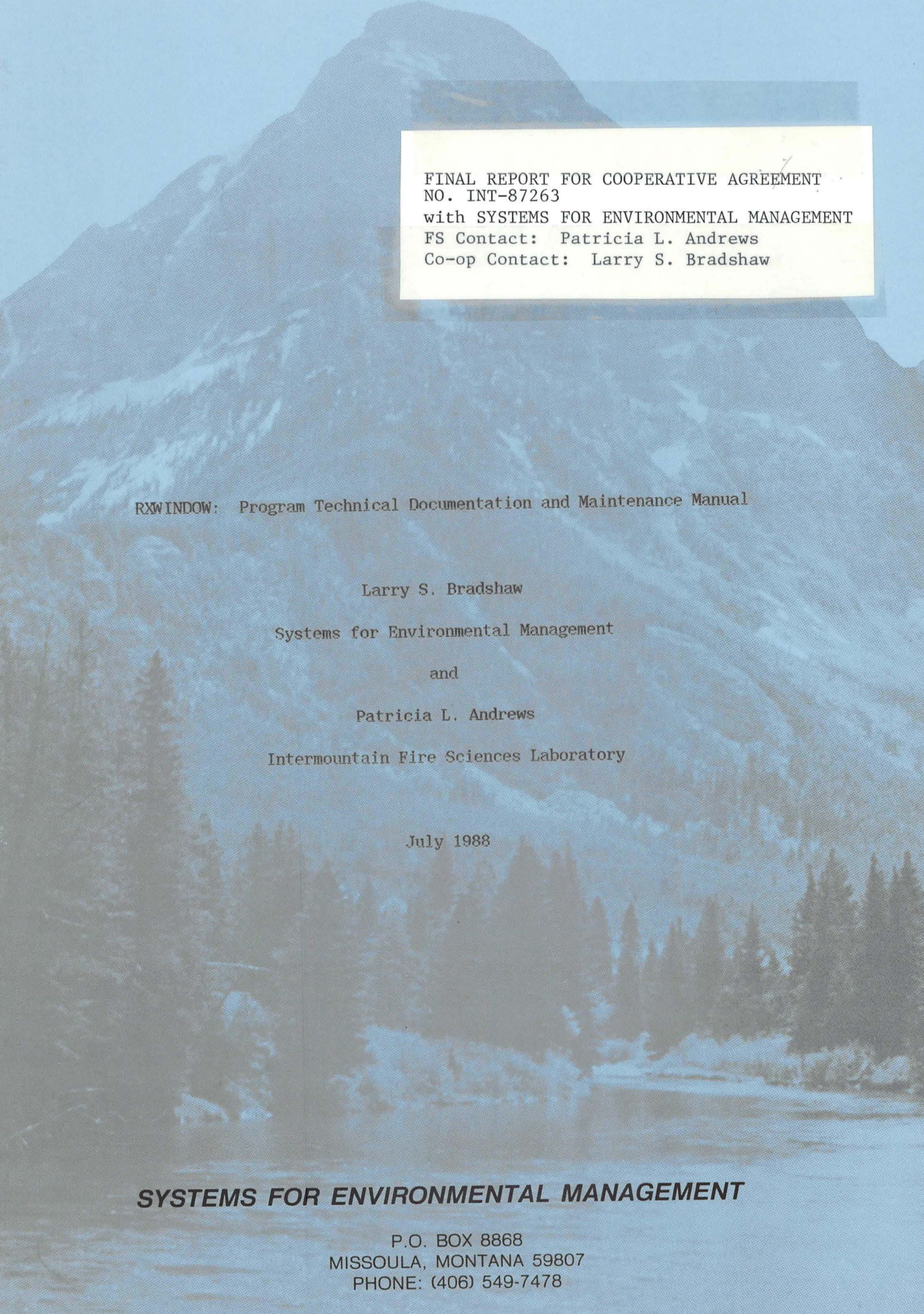


**Thesis/
Reports
Bradshaw,
L. S.**

**RXWINDOW: Program
Technical Documentation and
Maintenance Manual**



FINAL REPORT FOR COOPERATIVE AGREEMENT
NO. INT-87263
with SYSTEMS FOR ENVIRONMENTAL MANAGEMENT
FS Contact: Patricia L. Andrews
Co-op Contact: Larry S. Bradshaw

RXWINDOW: Program Technical Documentation and Maintenance Manual

Larry S. Bradshaw

Systems for Environmental Management

and

Patricia L. Andrews

Intermountain Fire Sciences Laboratory

July 1988

SYSTEMS FOR ENVIRONMENTAL MANAGEMENT

P.O. BOX 8868
MISSOULA, MONTANA 59807
PHONE: (406) 549-7478

RXWINDOW: Program Technical Documentation and Maintenance Manual

Larry S. Bradshaw

Systems for Environmental Management

and

Patricia L. Andrews

Intermountain Fire Sciences Laboratory

July 1988

Contents

Preface	1
Introduction	1
Language and Source Code	2
Compiling	
Linking	
Porting to Other Computers	3
Opening Console	
External File Names	
Execution Speeds	
Memory Requirements	
Files and File Structures	4
Unit Conversions	5
Changing Secondary System	
Changing Default Startup Units	
Unit Conversion Narrative	
Unit Conversion Factors and Functions	
Appendix A - Subroutine Hierarchy	A-1
Appendix B - Common Block Definitions	B-1
DISPLAY	
INOUT	
INOUTC	
SPREADCB	
SPCOM	
UNITS	
UNTCHR	
WNDCOM	
WNDCOMC	
Appendix C - Routine Documentation	C-1
Appendix D - Source Code List	D-1
DISPLAY.INC	
IOCOM.INC	
QSPREAD.INC	
SPCOM.INC	
UNITS.INC	
WNDCOM.INC	
RXWINDOW.F77	
RXWINDOW1.F77	
RXWINDOW2.F77	

Preface

This research was supported in part by funds provided by the U.S. Department of Agriculture, Forest Service, Intermountain Research Station.

This report is submitted in fulfillment of Section III, paragraph E of Cooperative aid agreement INT-87263 between Systems for Environmental Management (SEM), Missoula, Montana and the Intermountain Research Stations's, Intermountain Fire Sciences Laboratory (IFSL), Missoula, MT.

Introduction

Cooperative aid agreement INT-87263 was initiated to complete work on a prototype version of a fire prescription window system (Andrews and Bradshaw, 1987) developed under cooperative aid agreement 22-6-C-INT-038. The final version of the program, RXWINDOW has been installed on the IFSL Data General computer. This report serves as technical documentation of routines and variables comprising the system. Readers are referred to two other reports on the RXWINDOW system for further information as needed. They are:

1. Interface of fire behavior prediction and tree mortality models for application in the BEHAVE and RXWINDOW computer programs. Intermediate report to INT-87263, and
2. RXWINDOW: Fire prescription development program users manual. Final report to INT-87263.

Both reports are on file with the fire behavior research work unit (IFSL) and at SEM in Missoula.

Other referenced reports and publications that may be useful are:

1. BEHAVE: Fire behavior prediction and fuel modeling system- BURN subsystem. By Patrica L. Andrews (1986),
2. Technical documentation and maintenance of the BURN subsystem. By Patrica L. Andrews and Larry S. Bradshaw (1985),
3. BEHAVE: Fire behavior prediction and fuel modeling system- FUEL subsystem. By Burgan and Rothermel (1984),
4. A model for predicting tree mortality in western conifers. By Elizabeth Reinhardt and Kevin Ryan (1987), and
5. Documentation of modifications to programs in the BEHAVE fire behavior prediction and fuel modeling system to enable english or metric units. Larry S. Bradshaw, final report to contract 40-0357-7-302 between SEM and the IFSL, September 1987.

Since RXWINDOW will primarily be used within the BEHAVE arena, its design and user interface emulates other BEHAVE programs, particularly FIRE1 and FIRE2. It may use custom fuel models developed with BEHAVE's fuel model development programs, NEWMDL and TSTMDL with two restrictions:

1. The two fuel model concept is not allowed, and
2. All fuel models are considered static.

Language and Source Code

RXWINDOW is written in FORTRAN 77 and conforms to ANSI X3.9-1978 standards.
The program is contained in three source files:

RXWINDOW.F77 - MAIN and BLOCK DATA routines,
RXWINDOW1.F77 - Computational routines, and
RXWINDOW2.F77 - General utility routines.

Six "include" files contain all block common definitions:

DISPLAY.INC - Variables for displaying tables
IOCOM.INC - Variables for logical units and external file names
QPSREAD.INC - Variables for optimized fire behavior routine
SPCOM.INC - Variables for fuel model parameters
UNITS.INC - Array pointers and unit conversion parameters
WNDCOM.INC - General program wide variables and flags

All RXWINDOW files currently reside in the deflected drawer :WINDOW:PROGRAM: on
the IFSL DG MV/4000.

Compiling

Use the standard F77 command to compile a source code file:

F77 RX1

Linking

Use the standard F77LINK command to link compiled object files:

F77LINK RX RX1 RX2

Porting to Other Computers

The following topics may require attention for porting this program to other computers (including micros) or compiling with different versions of FORTRAN.

1. Opening console for read and write.

The Data General operating system uses logical units 5 and 10 for preconnection to the console. RXWINDOW initializes LU5=5 and LU6=10 for console read and writes. On systems with other preconnected logical units, LU5 and LU6 may be initialized as needed in BLOCK DATA. For systems without preconnection, two inactive (by comment) OPEN statements in the MAIN routine can be activated to open the console.

2. External file names.

Log and fuel model file names are limited to 12 characters in length.

3. Execution time.

Because of the extensive mathematical computations involved, execution times on microcomputers will be slow, particularly on units without math co-processors.

4. Memory Requirements

Block common /DISPLAY/ contains two large arrays that are of type INTEGER*2 to conserve required memory. These arrays are dimensioned with PARAMETER values:

- MXF - Number of weighted fuel moisture slots in moisture array
- MXP - Number of particle moisture slots in moisture array
- MX20 - Number of 20 foot wind slots in prescription array
- MXWD - Number of weighted dead moisture slots in prescription array
- MXS - Number of fire spread direction slots in prescription array
- MXI - Number of information items for each prescription array element

The default configuration requires about 270K total program memory. Using INTEGER*4 the same configuration requires 330K memory. Increasing MX20 to 51 and MXWD to 50 requires 356K with INTEGER*2 and 505K with INTEGER*4 data types.

Files and File Structures

External Files

Four external files used in RXWINDOW are summarized in the following table.

Name	Device	Unit	Open Notes	Close Notes
Input	Console	LU5	Preconnected when LU5 = 5 (DG)	Termination
Output	Console	LU6	Preconnected when LU6 = 10 (DG)	Termination
Log	Disk	LU4	Opened in LOGIT, user named	MAIN on QUIT
Fuel	Disk	LU7	Opened in CUSTOM, user named	MAIN on QUIT

RXWINDOW uses sequential formatted files for its two disk files. The user named log file contains display from commands LIST, RUN, LAST, REPLAY, ZOOM, and COMMENT while LOG is true. The user specified custom fuel model file contains parameters for fuel models developed with BEHAVE programs NEWMDL or TSTMDL and consists of one header record and any number of fuel model records:

Fuel Model File: Header Record Structure

Column(s)	Format	Contents
1 - 4	A4	Password
5 - 6	2X	Blank
7 - 78	18A4	File Description
79 - 79	1X	Blank
80 - 80	A1	Format Indicator

Fuel Model File: Fuel Model Record Structure

Column(s)	Format	Contents
1 - 2	I2	Fuel model number
3 - 3	F1.1	Wind reduction factor
4 - 35	8A4	Fuel model name
36 - 39	F4.2	1-h load, t/ac
40 - 43	F4.2	10-h load, t/ac
44 - 47	F4.2	100-h load, t/ac
48 - 51	F4.2	Live herbaceous load, t/ac
52 - 55	F4.2	Live woody load, t/ac
56 - 59	F4.2	Fuel bed depth, ft
60 - 64	F5.0	Heat content, btu/lb
65 - 66	F2.0	Moisture of extinction, %
67 - 70	F4.0	1-h surface area to volume ratio, 1/ft
71 - 74	F4.0	Live herbaceous surface area to volume ratio, 1/ft
75 - 78	F4.0	Live woody surface area to volume ratio, 1/ft
79 - 79	A1	Format indicator
80 - 80	I1	Static/Dynamic Flag (0=Static, 1=Dynamic)

Internal Files

RXWINDOW uses internal files to prepare output strings in subroutines INRX, SETRNG, SHOWML, SHOWMD, and SHOWRX.

Unit Conversions

RXWINDOW employs the same multi-unit concept available in the latest releases of the BEHAVE programs. Unit specification is controlled by four keywords:

METRIC	--	Select metric 'secondary' units,
ENGLISH	--	Select english 'base' units,
PERCENT	--	Select slope in percent,
DEGREES	--	Select slope in degrees,

two COMMON blocks:

/UNITS/	--	All new numeric data for unit conversion,
/UNTCHR/	--	All new character data for unit conversion,

one SUBROUTINE:

SETRNG	--	Sets valid range into a prompt string,
--------	----	--

and three FUNCTIONS:

CVRTU	--	Performs unit conversion on input and output values,
IEOS	--	Locates the end of a character string (last non blank),
SETUNT	--	Called by METRIC, ENGLISH, PERCENT, DEGREES keywords.

Changing The Secondary Units Systems

Effecting program wide changes in a variables' secondary units is simply a matter of changing the initialization of two arrays in BLOCK DATA for each variable you want to change, compiling and linking the program.

1. UNTLBL(I,J,K) -- Change UNTLBL(1,2,K) to the long secondary unit label
Up to 10 characters
-- Change UNTLBL(2,2,K) to the short secondary unit label
Up to 10 characters, preferably less than 6
2. CFAC(K) -- Change CFAC(K) to the conversion factor from base to
secondary units

Special Conditions: If the conversion requires a special algorithm it must be flagged and FUNCTION CVRTU modified to check for the flag value.

Currently CFAC(K) = -1. denotes temperature conversion
= -2. denotes slope conversion

K is the array element that holds information for the variable. See common block definitions for UNITS for values of k.

Changing The Defalut Start-up Units

Default start-up units are initialized in the BLOCK DATA routine. Four variables define default start-up units:

Variable	Start-up In Base Units	Start-up In Secondary Units
METRIC	.FALSE.	.TRUE.
IUNT	1	2
PERCENT	.TRUE.	.FALSE.
ISLP	1	2

To initiate these changes, change the initializations BLOCK DATA, compile and link the program.

Unit Conversion Narrative

Unit sensitive input solicitations are constructed in three parts--a prompt, a unit label, and a valid range into character string, STR:

```
|----- Prompt ----- |-- Unit --|-- Range -|  
STR = "7. 20-Foot Wind Speed, Kmeters/H (0-125)  "
```

Input prompts and valid range limits (in base units) originate in INRX calls to READIT or READQ. For input of 20-Foot Wind Speed, '20-Foot' is replaced with '6-Meter' for metric units.

READIT and READQ intercept the original prompt and base unit range. If the prompt is not unit sensitive, it is passed unchanged to ASK2. If it is unit sensitive, range limits are converted if needed and the prompt is reconstructed to reflect the active units before being sent to ASK2.

***** I M P O R T A N T *****

In order for any input prompt, list variable, or output variable to be processed as unit sensitive, the character string identifying it must contain the name followed by a comma and a blank (' , '). For example, the string

"Forward rate of spread, " is processed as unit sensitive, while

"Crown ratio" is not.

The occurrence of this substring (,) identifies the item as unit sensitive and marks the place in the string to position the correct unit label.

CHECK, an input support subroutine (called by READIT and READQ) converts metric input to base units and assign it a proper position in the VK working array, or to a scaler variable. The variable pointer value (NVAR) passed through from INRX routines has three conditions:

- NVAR > 0 : Assign value to VK array at position NVAR
- NVAR = 0 : Assign value to scaler, but value is not unit sensitive.
- NVAR < 0 : Assign value to scaler, absolute value of NVAR points to position in CFAC() array with correct conversion factor.

Unit sensitive list variable labels are constructed from two parts--a variable name plus a unit label into character string, STR:

```
|----- Name -----|--- Unit ---|
STR = "7. 20-Foot Wind Speed, Kmeters/H "
```

Labels originate in LIRX calls to the listing utility, LISTIT. For listing 20-Foot Wind Speed, '20-Foot' is replaced with '6-Meter' for metric units.

LISTIT converts base values to active units for listing and intercepts the original variable label string. If the variable is not unit sensitive, it is unchanged before listing. If it is unit sensitive it is reconstructed to reflect the active units.

Unit sensitive output variable labels are constructed from two parts--an output variable name plus a unit label into character string, STR:

```
|----- Name -----|--- Unit ---|
STR = "Forward Rate of Spread, meters/min "
```

Output variable labels originate in the SHOW series of routines where base unit values are converted to active units just prior to display.

Subroutine CUSTOM displays the active unit labels when displaying the values of a fuel model which are converted from base to active units just prior to display.

Unit Conversion Factors and Functions

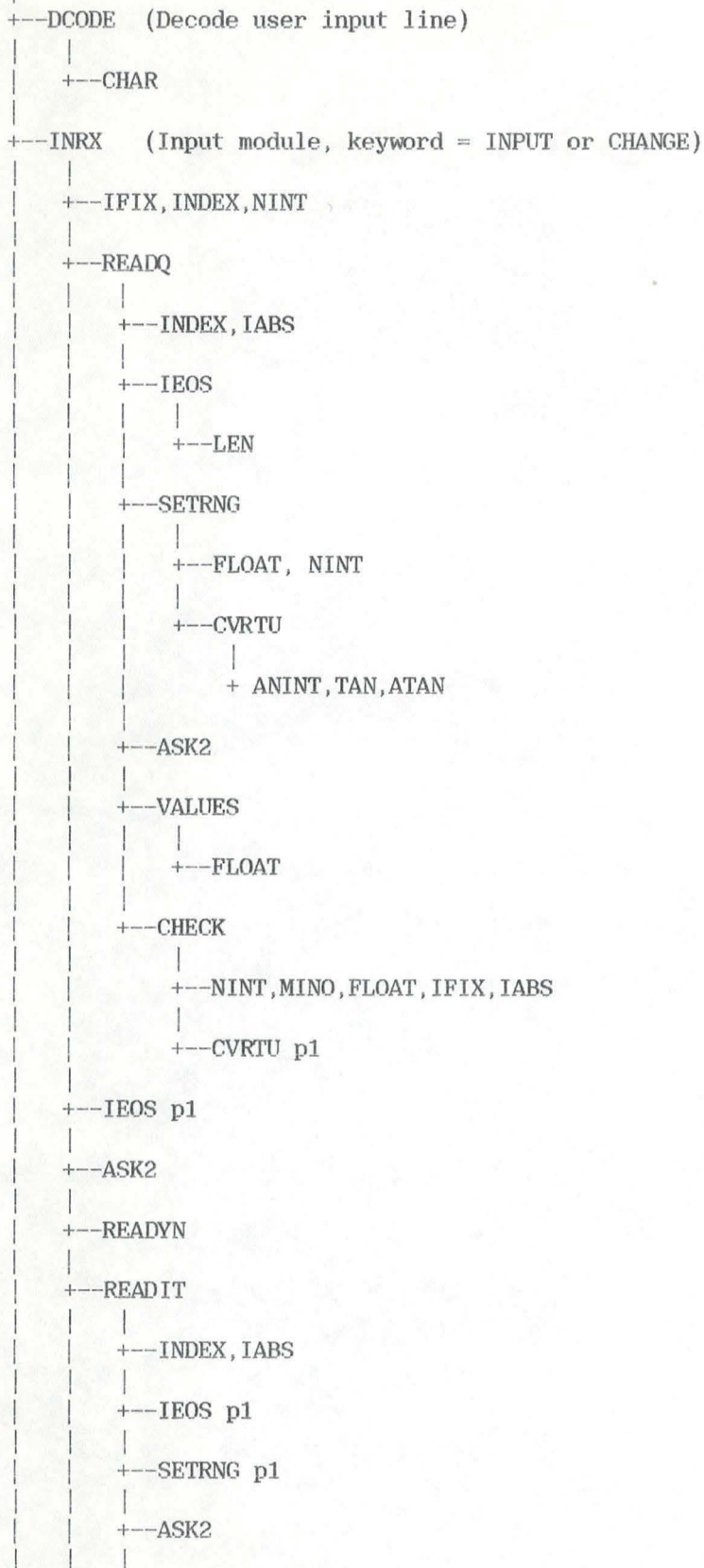
Parameter	Base Units	Secondary Units	Conversion Factor
Length	Feet (Ft)	Meters (M)	0.305
Length	Inches	Centimeters	2.54
Spread Rate	Chains/Hour	Meters/Min	0.335
Wind Speed	Miles/Hour	Kmeters/Hour	1.609
Heat/Area	Btu/Sq.Ft.	KJoules/Sq.Meter	11.357
Fireline Intensity	Btu/Ft-Sec	Kwatt/Meter	3.461
Reaction Intensity	Btu/Sq.Ft.Min	KWatts/Sq.Meter	0.189
Surface Area/Volume	1/Feet	1/Centimeter	.033
Fuel Load	Tons/Acre	Metric Tons/Hectare	2.242
Fuel Heat Content	Btu/Pound	Joules/Gram	2.324
Fuel Bed Depth	Feet	Centimeters	30.48

Special Functions:

Temperature (-1): Fahrenheit to Celcius	C = .555*(F-32)
: Celcius to Fahrenheit	F = 1.8*C + 32
Slope (-2) : Percent to Degrees	D = ATAN(percent*0.01)*57.2727
: Degrees to Percent	D = TAN(degrees*0.01746)*100.

Subroutine Hierarchy Chart

RXWINDOW (MAIN)



```

|--VALUES p1
|--CHECK p1
+--INFUEL
|--WHAT
|   |--VALUES p1
|   |--CHECK p1
+--CHKMOD
|   |--READYN
|   |--READIT p1
|   |--MFLAG
+--READYN
+--CUSTOM
|   |--READYN
|   |--WAITE
|   |--READIT p1
|   |--CVRTU p1
+--READIT p1
+--WHAT p2
+--SETWAF
|   |--NINT
|   |--READIT p1
+--SCORLIM
|   |--LOG,MAX,MIN,ABS

```

```

|
|--CVRTU p1
|
|--SETUP
|
|--QSPREAD
|   |
|   |--EXP
|
|--MORT
|   |
|   |--MAX,EXP,MIN
|
|--LIRX (List module, keyword = LIST)
|
|--NINT
|
|--LISTIT
|   |
|   |--IFIX,FLOAT,INDEX,LEN,NINT
|   |
|   |--CVRTU p1
|   |
|   |--IEOS p1
|
|--WAITE
|
|--LIFUEL
|   |
|   |--INDEX
|
|--IEOS p1
|
|--CARX (Calculate prescription window and moisture tables, keyword = RUN)
|
|--NINT,FLOAT
|
|--SETMOI (ENTRY in SETUP)
|
|--SETUP
|
|--QSPREAD p3
|
|--TESTRX
|   |
|   |--FLOAT, NINT
|   |
|   |--SETMOI (ENTRY in SETUP)
|   |
|   |--QSPREAD p3
|   |
|   |--VECTOR
|   |   |
|   |   |--COS,SIN,SQRT,ASIN,ABS

```

```

+--FLANK
|
+--SQRT,ABS,COS
|
+--FBTEST
|
+--NINT
|
+--MORT p3
|
+--TRANGE
|
+--WINNER
|
+--MAXO,MINO
|
+--WAITE
|
+--SHOWRX
|
+--FLOAT,NINT
|
+--IEOS p1
|
+--WAITE
|
+--CVRTU p1
|
+--CONTRL
|
+--DCODE p1
|
+--LIRX p3
|
+--ZOOMRX
|
+--FLOAT,NINT
|
+--READIT p1
|
+--SHOWRX p4
|
+--ZOOMMD
|
+--NINT,FLOAT
|
+--READIT p1
|
+--SHOWMD
|
+--IEOS p1
|
+--WAITE

```

```

+--ZOOMML
|
|   +--NINT
|   |
|   +--READIT p1
|   |
|   +--SHOWML
|   |
|   |   +--IEOS p1
|   |   |
|   |   +--WAITE
|
+--REPLRX
|
|   +--SHOWRX p4
|
+--REPLMD
|
|   +--SHOWMD p4
|
+--REPLML
|
|   +--SHOWML p5
|
+--HLPWIN
|
|   +--KEYO
|   |
|   +--WAITE
|   |
|   +--KEYALL
|
+--KEYO
|
+--KEYALL
|
+--STATUS
|
+--COMENT
|
+--KEYFLG
|
|   +--LOGIT
|   |
|   |   +--IEOS p1
|
+--SETUNT
|
+--READYN

```

```

|
|  +---CAMOIS  (Calculate and display moisture tables)
|  |
|  |  +---NINT,FLOAT,MINO,MAXO,ZERO
|  |  |
|  |  |  +---SETMOI (ENTRY in SETUP)
|  |  |  |
|  |  |  |  +---QSPREAD p3
|  |  |  |  |
|  |  |  |  |  +---SHOWMD p4
|  |  |  |  |  |
|  |  |  |  |  |  +---CONTRL p4
|  |  |  |  |  |  |
|  |  |  |  |  |  |  +---SHOWML p5
|  |
|  |  +---LAWIND (ENTRY in CARX for last prescription table)
|  |
|  |  +---LADEAD (ENTRY in CAMOIS for last dead moisture table)
|  |
|  |  +---LALIVE (ENTRY in CAMOIS for last live moisture table)
|  |
|  |  +---HLPWIN p5 (Help system, keyword = HELP)
|  |
|  |  +---KEY0   (List keywords, keyword = KEY)
|  |
|  |  +---KEYALL (List more keywords, keyword = KEY)
|  |
|  |  +---STATUS (Display status, keyword = STATUS)
|  |
|  |  +---COMENT p5 (Enter comment to log file, keyword = COMMENT)
|  |
|  |  +---READYN
|  |
|  |  +---KEYFLG p5 (Test for flag setting keywords LOG, PAUSE, WORDY)
|  |
|  |  +---SETUNT   (Test for unit keywords METRIC, ENGLISH, DEGREES, PERCENT)

```

Subroutine Hierarchy Index

Routine	Definition Filename	Routine Description	Tree Diagram Reference Page
ABS	Intrinsic	Absoulte value (real)	3 4
ANINT	Intrinsic	Nearest whole number	1
ASIN	Intrinsic	Arcsin	3
ATAN	Intrinsic	Arctangent	1
CHAR	Intrinsic	Integer to ASCII	2
EXP	Intrinsic	Exponent	3
FLOAT	Intrinsic	Integer to Real	1 3 4 5 6
IABS	Intrinsic	Absolute (integer)	1 2
ICHAR	Intrinsic	ASCII to integer	2
IFIX	Intrinsic	Truncate	1 2 3
INDEX	Intrinsic	String position	1 2 3
LEN	Intrinsic	String length	1 3
LOG	Intrinsic	Log	3
MAXO	Intrinsic	Maximum (integer)	4 6
MAX1	Intrinsic	Maximum (real)	3
MINO	Intrinsic	Minimum (integer)	1 4 6
MIN1	Intrinsic	Minimum (real)	3
NINT	Intrinsic	Nearest Integer	1 2 3 4 5 6
SIN	Intrinsic	Sin	4
SQRT	Intrinsic	Square root	4
TAN	Intrinsic	Tangent	1
RXWINDOW	RXWINDOW.F77	Keyword based program control	1
ASK2	RXWINDOW2.F77	Ask a question	1 2
CAMOIS	RXWINDOW1.F77	Calculate Moisture Tables	6
CARX	RXWINDOW1.F77	Calculate RX Table	3
CHECK	RXWINDOW2.F77	Check Input Syntax	1 2
CHKMOD	RXWINDOW2.F77	Check Fuel Model Specs	1 2
COMENT	RXWINDOW2.F77	Get and log a comment	6 7
CONTRL	RXWINDOW1.F77	Control from CARX/CAMOIS	4 6
CUSTOM	RXWINDOW2.F77	Load custom fuel model	2
CVRTU	RXWINDOW2.F77	Perform unit conversion	1 3 4
DCODE	RXWINDOW2.F77	Get KEY and ARG from input	2 4
FBTEST	RXWINDOW1.F77	Test behavior to constraints	4
FLANK	RXWINDOW2.F77	Fire behavior in given direct.	4
HLPWIN	RXWINDOW2.F77	Help system	5 6
IEOS	RXWINDOW2.F77	End of string position	1 2 3 4 5 6
INFUEL	RXWINDOW2.F77	Fuel model input	2
INRX	RXWINDOW1.F77	Program input module	2
KEYO	RXWINDOW2.F77	Module keyword list	5 6
KEYALL	RXWINDOW2.F77	Flag keyword list	5 6
KEYFLG	RXWINDOW2.F77	Checks for flag keys	5 6
LADEAD	RXWINDOW1.F77	Entry in CAMOIS for last dead	6
LALIVE	RXWINDOW1.F77	Entry in CAMOIS for last live	6
LAWIND	RXWINDOW1.F77	Entry in CARX for last RX	6
LIFUEL	RXWINDOW2.F77	List fuel model	3
LISTIT	RXWINDOW1.F77	List individual variable	3
LIRX	RXWINDOW1.F77	List input	3 4
LOGIT	RXWINDOW2.F77	Control log file	6
MFLAG	RXWINDOW2.F77	Set moisture flags	2
MORT	RXWINDOW1.F77	Get mortality	3 4

QSPREAD	RXWINDOW2.F77	Fire behavior module	3	4	6	
READIT	RXWINDOW2.F77	Ask question, check input	1	2	3	5
READQ	RXWINDOW2.F77	Same as READIT, quit allowed	2			
READYN	RXWINDOW2.F77	Get yes/no response	1	2	6	
REPLMD	RXWINDOW2.F77	Replay last dead table	5			
REPLML	RXWINDOW2.F77	Replay last live table	5			
REPLRX	RXWINDOW2.F77	Replay last RX table	5			
SCORLIM	RXWINDOW2.F77	Gets scorch, given mortality	3			
SETMOI	RXWINDOW2.F77	Entry in SETUP for moisture	3	4	6	
SETRNG	RXWINDOW2.F77	Sets string with range values	1	2		
SETUNT	RXWINDOW2.F77	Control for unit keywords	5	6		
SETUP	RXWINDOW2.F77	Sets up variables for QSPREAD	1	3	4	
SETWAF	RXWINDOW2.F77	Sets wind adjustment factor	3			
SHOWMD	RXWINDOW1.F77	Shows dead moisture table	5	6		
SHOWML	RXWINDOW1.F77	Shows live moisture table	5	6		
SHOWRX	RXWINDOW1.F77	Shows prescription table	4	5		
STATUS	RXWINDOW2.F77	Displays current status	5	6		
TESTRX	RXWINDOW1.F77	Controls prescription testing	4			
TRANGE	RXWINDOW2.F77	Computes scorch temp ranges	4			
VALUES	RXWINDOW2.F77	Decodes input values	1	2		
VECTOR	RXWINDOW2.F77	Gets direction of max spread	4			
WAITE	RXWINDOW2.F77	Pauses screen display	2	3	4	5 6
WHAT	RXWINDOW2.F77	Gets input after question	2	3		
WINNER	RXWINDOW1.F77	Updates INRX array	4			
ZERO	RXWINDOW2.F77	Initializes MSD array	6			
ZOOMMD	RXWINDOW2.F77	Zooms dead moisture table	5			
ZOOMML	RXWINDOW2.F77	Zooms live moisture table	5			
ZOOMRX	RXWINDOW2.F77	Zooms prescription table	4			

COMMON Block Definitions

```

C...INCLUDE FILE NAME      :  DISPLAY.INC
C...BLOCK COMMONS DEFINED:  /DISPLAY/
C...
C... COMMON BLOCK /DISPLAY/ FOR VARIABLES USED IN WINDOW DISPLAY MODULES
C...
C... MOST OF THESE VARIABLES ARE SET IN CARX, CAMOIS, AND WINNER
C...
      PARAMETER (MXF=50,MXP=50,MX20=31,MXWD=30,MXS=3,MXI=9)
C...
      COMMON /DISPLAY/ LWMN,LWMX,LWST,LDMN,LDMX,LDST,ICOL(10,2),
1  LDSTRT,LDSTOP,LDSTEP,L1STRT,L1STOP,L1STEP,L1NUM,
2  LLSTRT,LLSTOP,LLSTEP,LWSTRT,LWSTOP,LWSTEP,LWNUM,
3  LOWI,MXW,MAXWI,LOD,MXD,MXLV,MNLV,
4  MAXMIN(MX20),MAXMAX(MX20),MAXWND(MX20),
5  MSD(MXF,MXP,2),INRX(MX20,MXWD,MXS,MXI),NONEIN
C...
      INTEGER*2 MSD,INRX
      LOGICAL NONEIN
C...
C... DATA DICTIONARY FOR COMMON BLOCK /DISPLAY/
C...
C  MXF      :  Dimension parameter for weighted fuel moisture slots (see MSD)
C  MXP      :  Dimension parameter individual particle moistures      "
C  MX20     :  Dimension parameter for 20 foot wind slots             (see INRX)
C  MXWD     :  Dimension parameter for weighted dead slots           "
C  MXS      :  Dimension parameter for fire spread direction slots    "
C  MXI      :  Dimension parameter for prescription utiltiy slots      "
C
C  LWMN     :  MINIMUM 20 FOOT WIND FOR CURRENT WHOLE TABLE
C  LWMX     :  MAXIMUM 20 FOOT WIND FOR CURRENT WHOLE TABLE
C  LWST     :  WIND STEP SIZE FOR CURRENT WHOLE TABLE
C  LDMN     :  MINIMUM WEIGHTED DEAD FM FOR CURRENT WHOLE TABLE
C  LDMX     :  MAXIMUM WEIGHTED DEAD FM FOR CURRENT WHOLE TABLE
C  LDST     :  WEIGHTED DEAD FM STEP SIZE FOR CURRENT WHOLE TABLE
C  ICOL()   :  ROW OUTPUT VALUE RANGES FOR 10 COLUMNS (1=MIN,2=MAX)
C
C  LDSTRT   :  LAST DISPLAY TABLE'S WEIGHTED DEAD FM START VALUE
C  LDSTOP   :  LAST DISPLAY TABLE'S WEIGHTED DEAD FM STOP VALUE
C  LDSTEP   :  LAST DISPLAY TABLE'S WEIGHTED DEAD FM STEP VALUE
C
C  L1STRT   :  LAST DISPLAY TABLE'S 1-HR DEAD FM START VALUE
C  L1STOP   :  LAST DISPLAY TABLE'S 1-HR DEAD FM STOP VALUE
C  L1STEP   :  LAST DISPLAY TABLE'S 1-HR DEAD FM STEP VALUE
C  L1NUM    :  NUMBER OF STEPS ALLOWED IN TABLE
C
C  LLSTRT   :  LAST DISPLAY TABLE'S WEIGHTED LIVE FM START VALUE
C  LLSTOP   :  LAST DISPLAY TABLE'S WEIGHTED LIVE FM STOP VALUE
C  LLSTEP   :  LAST DISPLAY TABLE'S WEIGHTED LIVE FM STEP VALUE
C
C  LWSTRT   :  LAST DISPLAY TABLE'S WIND START VALUE
C  LWSTOP   :  LAST DISPLAY TABLE'S WIND STOP VALUE
C  LWSTEP   :  LAST DISPLAY TABLE'S WIND STEP VALUE
C  LWNUM    :  NUMBER OF STEPS ALLOWED IN TABLE
C
C  LOD      :  MINIMUM DEAD FUEL MOISTURE IN PESCRPTION

```

C MXD : MAXIMUM DEAD FUEL MOISTURE IN PRESCRIPTION
C LOWI : MINIMUM 20 FOOT WIND IN PRESCRIPTION
C MXW : MAXIMUM 20 FOOT WIND IN PRESCRIPTION
C MXLV : MAXIMUM WEIGHTED LIVE FUEL MOISTURE IN OUTPUT TABLE
C MNLV : MINIMUM WEIGHTED LIVE FUEL MOISTURE IN OUTPUT TABLE

C MAXMIN(): HIGHEST MINIMUM LIVE FUEL MOIST FOR A GIVEN 20 FOOT WIND
C MAXMAX(): HIGHEST MAXIMUM LIVE FUEL MOIST FOR A GIVEN 20 FOOT WIND
C MAXWND(): MAXIMUM 20 FOOT WIND FOR ENTIRE PRESCRIPTION RANGE

C MSD(I,J,K): FOR A GIVEN WEIGHTED DEAD MOISTURE CONTENT (I) AND 1-HOUR
C FM (J), K=1 IS MINIMUM ALLOWED 10-HOUR FM AND K=2 IS
C MAXIMUM ALLOWED 10-HOUR FM. IF THE USER HAS SELECTED
C 10-HOUR ACROSS THE TOP OF THE TABLE, THEN THE VALUES
C IN THE (J) AND (K) SLOTS ARE REVERSED.

C FOR LIVE TABLES, I IS WEIGHTED LIVE, J IS WOODY, AND K IS
C HERBACEOUS. IF USER HAS SELECTED WOODY AS TABLE VALUE, J
C AND K ARE REVERSED.

C INRX(I,J,K,L) : I = WIND SPEED
C J = WEIGHTED DEAD FUEL MOISTURE
C K = FIRE SPREAD DIRECTION (HEAD,FLANK,REAR)
C L = POINTER VALUES:
C 1 = KOUNTER OF NUMBER IN RX
C 2 = MIN WEIGHTED LIVE FM RX
C 3 = MAX WEIGHTED LIVE FM RX
C 4 = BEGIN WIND DIRECTION IN RX
C 5 = END WIND DIRECTION IN RX
C 6 = MIN TEMP FOR SCORCH IN RX
C 7 = MAX TEMP FOR SCORCH IN RX
C 8 = MIN TABLE FIRE BEHAVIOR VALUE IN CELL
C 9 = MAX TABLE FIRE BEHAVIOR VALUE IN CELL

C NONEIN : .TRUE. = CURRENT RX HAS NO SOLUTION

COMMON Block Definitions

```
C...INCLUDE FILE NAME      :  IOCOM.INC
C...BLOCK COMMONS DEFINED:  /INOUT/,/INOUTC/
C...
C... COMMON BLOCK FOR ALL IO NEEDS (INITIALIZED IN BLOCK DATA)
C...
      COMMON /INOUT/ LU4,LU5,LU6,LU7
      COMMON /INOUTC/ LOGFIL,FILNAM
      CHARACTER LOGFIL*8,FILNAM*12
C...
C... DATA DICTIONARY FOR /INOUT/
C
C      LU4 : LOGICAL UNIT FOR LOG FILE
C      LU5 : LOGICAL UNIT FOR CONSOLE WRITE
C      LU6 : LOGICAL UNIT FOR CONSOLE READ
C      LU7 : LOGICAL UNIT FOR CUSTOM FUEL MODEL FILE
C...
C... DATA DICTIONARY FOR /INOUTC/
C
C      LOGFIL : CHARACTER*8, NAME OF LOG FILE
C      FILNAM : CHARACTER*12, NAME OF FUEL FILE
C
C
```

COMMON Block Definitions

```

C...INCLUDE FILE NAME      : QSPREAD.INC
C...BLOCK COMMONS DEFINED: /SPREADCB/
C
C  MOST VARIABLES SET IN SETUP, QSPREAD, CARX, AND CAMOIS
C
C PARAMETERS DECLARATION
C
      INTEGER NCLASS,NCATEG,DEAD,LIVE,HR1,HR10,HR100
      PARAMETER (NCLASS=3,NCATEG=2,DEAD=1,LIVE=2,HR1=1,HR10=2,HR100=3)
C
C /SPREADCB/ VARIABLE TYPES
C
C...NUMBER FUEL CLASSES
      INTEGER NFC
C...FUEL INPUTS
      REAL FUELLOAD,FUELSAVR,FUELSTOT,FUELSEFF,FUELHEAT,FUELDENS,
      + FUELDEPTH,FUELMEXT
C...ENVIRONMENTAL INPUTS
      REAL FUELMOIS,SLOPE,SDIR,WIND,WDIR,WMAX
C...INTERMEDIATE VARIABLES
      REAL FUELAREA,FUELG,FCATAREA,FCATLOAD,FCATSAVR,FCATETAS,FCATSEFF,
      + FCATHEAT,BETAOP,AA,GAMMA,GAMMAX,C,E
C...DERIVED FUEL BED CONSTANTS
      REAL FUELF,FUELMWF,FCATF,PRI,FINED,WLIVE,SIGMA,BETA,RATIO,RHOB,
      + XI,KSLOPE,B,KWIND
C...DERIVED ENVIRONMENTAL INTERMEDIATES
      REAL QIG,FCATMOIS,FCATETAM,FCATMEXT,WMFD,FDMOIS
C...DERIVED ENVIRONMENTAL CONSTANTS
      REAL RBQIG,XIR,PHIW,PHIS,PHIEW,EWIND,LWIND,WLIM
C...FIRE BEHAVIOR RESULTS
      REAL RATEO,RATE,HPUA,BYRAM,FLAME,SCOR77
C-----
C /SPREADCB/ OPTIMIZED SPREAD COMMON BLOCK
C          CONTAINS 2 INTEGER*4 AND 132 REALS*4 (536 BYTES)
C          AS DECLARED AND DEFINED ABOVE
C-----
COMMON /SPREADCB/
+   NFC(NCATEG),
+   FUELLOAD(NCLASS,NCATEG),FUELSAVR(NCLASS,NCATEG),
+   FUELSTOT(NCLASS,NCATEG),FUELSEFF(NCLASS,NCATEG),
+   FUELHEAT(NCLASS,NCATEG),FUELDENS(NCLASS,NCATEG),
+   FUELDEPTH,FUELMEXT,
+   FUELMOIS(NCLASS,NCATEG),SLOPE,SDIR,WIND,WDIR,WMAX,
+   FUELAREA(NCLASS,NCATEG),FUELG(NCLASS,NCATEG),FCATAREA(NCATEG),
+   FCATLOAD(NCATEG),FCATSAVR(NCATEG),FCATETAS(NCATEG),
+   FCATSEFF(NCATEG),FCATHEAT(NCATEG),BETAOP,AA,GAMMA,GAMMAX,
+   FUELF(NCLASS,NCATEG),FUELMWF(NCLASS,NCATEG),FCATF(NCATEG),
+   PRI(NCATEG),FINED,WLIVE,SIGMA,BETA,RATIO,RHOB,XI,KSLOPE,
+   B,C,E,KWIND,
+   QIG(NCLASS,NCATEG),FCATMOIS(NCATEG),FCATETAM(NCATEG),
+   FCATMEXT(NCATEG),WMFD,FDMOIS,
+   RBQIG,XIR,PHIW,PHIS,PHIEW,EWIND,LWIND,WLIM,
+   RATEO,RATE,HPUA,BYRAM,FLAME,SCOR77

```

COMMON Block Definitions

```
C...INCLUDE FILE NAME      :  SPCOM.INC
C...BLOCK COMMONS DEFINED:  /SPCOM/
C...
C... COMMON BLOCK DEFINITION FOR /SPCOM/ - SPREAD VARIABLES
C...
C... VARIABLES INITIALIZED IN BLOCK DATA, SETUP, CHKMOD, CUSTOM

      COMMON /SPCOM/ XLOAD,SIG,STOT,SEFF,HEAT,XMOIS,DENS,NCLAS
      DIMENSION XLOAD(2,3),SIG(2,3),STOT(2,3),SEFF(2,3),HEAT(2,3),
-      XMOIS(2,3),DENS(2,3),NCLAS(2)

C
C... DATA DICTIONARY FOR COMMON BLOCK /SPCOM/
C
C      XLOAD  :  FUEL LOADS
C      SIG    :  SURFACE AREA TO VOLUME RATIO
C      STOT   :  TOTAL MINERAL CONTENT
C      SEFF   :  EFFECTIVE MINERAL CONTENT
C      HEAT   :  FUEL HEAT CONTENT
C      XMOIS  :  FUEL EXTINCTION MOISTURE
C      DENS   :  FUEL PARTICLE DENSITY
C      NCLASS :  # FUEL SIZE CLASS/COMPONENT
C...
```

COMMON Block Definitions

```

C...INCLUDE FILE NAME      : UNITS.INC
C...BLOCK COMMONS DEFINED: /UNITS/,/UNTCHR/
C...
C... COMMON BLOCK INCLUDE FILE (UNITS.COM) FOR UNIT CONVERSIONS AND
C... VARIABLE NAME DEFINITIONS
C...
C... ALL VALUES INITIALIZED IN BLOCK DATA
C...
C... /UNITS/ CONTAINS POINTERS, LOGICAL FLAGS, RANGE AND CONVERSION ARRAYS
C...
      LOGICAL METRIC,TOBASE,NOT2BA,PERCENT
      COMMON /UNITS/ IUNT,IBAS,IMET,ISLP,IPCT,IDEG,LONG,ISHT,LIML,LIMH,
A  RNG(2,33),CFAC(33),METRIC,TOBASE,NOT2BA,PERCENT,
B  IROS,IHUA,IFLE,IFLI,IRIN,ISCO,IMOR,
C  IFUL,IWAF,ISLO,IREL,ISPC,IDBH,IHGT,ICRN,
D  I1HR,I10H,IHRB,IWDY,IW20,IWMD,IWDD,ISDD,
E  IW20B,I1DW,I1MD,IWGHT,IWDYL,IWLV,ITMP,
F  IUN1,IUN2,NON
C...
C... /UNTCHR/ CONTAINS CHARACTER CONSTANTS
      CHARACTER UNTLBL*10,STR*70,RNGSTR*20,VARLBL*40
      COMMON /UNTCHR/ UNTLBL(2,2,33),VARLBL(30),STR,RNGSTR
C...
C... DATA DICTIONARY FOR /UNITS/
C... /UNITS/ CONSISTS PRIMARILY OF POINTERS TO ARRAY SLOTS FOR ALL
C... DISPLAYABLE VARIABLES
C      IUNT : ACTIVE UNIT SET (1=BASE,2=METRIC)
C      IBAS : BASE UNIT POINTER (1)
C      IMET : METRIC UNIT POINTER (2)
C      ISLP : ACTIVE SLOPE UNITS (1=PERCENT,2=DEGREES)
C      IPCT : SLOPE IN PERCENT UNIT POINTER (1)
C      IDEG : SLOPE IN DEGREES UNIT POINTER (2)
C      LONG : POINTER TO LONG UNITS LABELS (1)
C      ISHT : POINTER TO SHORT UNITS LABELS (2)
C      LIML : POINTER TO INPUT VALUE LOW RANGE VALUE (1)
C      LIMH : POINTER TO INPUT VALUE HIGH RANGE VALUE (2)
C      RNG() : ALLOWABLE INPUT RANGE FOR A GIVEN VARIABLE

C      CFAC() : CONVERSION FACTOR FOR A GIVEN VARIABLE
C      METRIC : METRIC UNITS WHEN .T.

C      TOBASE : CONVERT INPUT TO BASE VALUE WHEN TRUE
C      NOT2BA : DO NOT CONVERT INPUT TO BASE WHEN TRUE
C      PERCENT: SLOPE UNITS IN PERCENT WHEN TRUE

C      IROS : POINTER TO RATE OF SPREAD (1)
C      IHUA : POINTER TO HEAT PER UNIT AREA (2)
C      IFLE : POINTER TO FLAME LENGTH (3)
C      IFLI : POINTER TO FIRELINE INTENSITY (4)
C      IRIN : POINTER TO REACTION INTENSITY (5)
C      ISCO : POINTER TO SCORCH HEIGHT (6)
C      IMOR : POINTER TO MORTALITY (7)

C      IFUL : POINTER TO FUEL MODEL (8)
C      IWAF : POINTER TO WIND ADJUSTMENT FACTOR (9)

```

C ISLO : POINTER TO SLOPE (10)
C IREL : POINTER TO 10 TO 100 HOUR RELATION (11)
C ISPC : POINTER TO SPECIES (12)
C IDBH : POINTER TO DIAMETER BREAST HEIGHT (13)
C IHGT : POINTER TO TREE HEIGHT (14)
C ICRN : POINTER TO CROWN RATIO (15)

C I1HR : POINTER TO 1 HOUR MOISTURE (16)
C I10H : POINTER TO 10 HOUR MOISTURE (17)
C IHRB : POINTER TO HERBACEOUS MOISTURE (18)
C IWDY : POINTER TO WOODY MOISTURE (19)
C IW20 : POINTER TO 20 FOOT WIND (20)
C IWMD : POINTER TO MIDFLAME WIND (21)
C IWDD : POINTER TO WIND DIRECTION (22)
C ISDD : POINTER TO SPREAD DIRECTION (23)

C IW20B : POINTER TO 20 FOOT FROM ZOOM ROUTINES (24)
C I1DW : POINTER TO 1 HOUR DEAD FROM ZOOM ROUTINES (25)
C I1MD : POINTER TO 1 HOUR DEAD FROM ZOOM ROUTINES (26)
C IWGHT : POINTER TO WEIGHTED DEAD FROM ZOOM ROUTINES (27)
C IWDYL : POINTER TO LIVE HERB MOISTURE FROM ZOOM ROUTINES (28)
C IWLVL : POINTER TO WEIGHTED LIVE FROM ZOOM ROUTINES (29)

C ITMP : POINTER TO AIR TEMPERATURE (30)

C IUN1 : POINTER TO UNUSED 1 (31)
C IUN2 : POINTER TO UNUSED 2 (32)
C NON : POINTER TO NO UNITS (33)

C... DATA DICTIONARY FOR COMMON BLOCK /UNTCHR/

C
C UNTLBL(I,J,K) : UNIT LABELS -- I = BASE/METRIC
C J = LONG/SHORT
C K = VARIABLE (SEE ABOVE)
C VARLBL(K) : VARIABLE LABEL FOR VARIABLE K (AS DEFINED ABOVE)
C STR : UTILITY CHARACTER STRING
C RNGSTR : STRING TO HOLD RANGE FOR PROMPTS WITH RANGES
C

COMMON Block Definitions

```

C...INCLUDE FILE NAME      : WNDCOM.INC
C...BLOCK COMMONS DEFINED: /WNDCOM/, /WDCOMC/
C...
C... COMMON BLOCK DEFINITIONS FOR WNDCOM.INC
C...
COMMON /WNDCOM/ VK(33,3),PMOD(15,15),NMOD(15,8),NEEDED(30),YEP,
A  W20,WAF,VERSN,NTWO,MODL,WMID,LIMITED(25),LAWOK,LADOK,LALOK,
B  PSLOP,FM(5),OK,WORDY,PAUSE,MOI(5),TABHB,TAB10,LOG,LOGOK,NOPE,
C  TWO,FILFLG,CHANGE,FIX,YES,LM1,LM2,REL100,IEXPOS,ITFB,QUIT
C...
LOGICAL OK,WORDY,PAUSE,MOI,TABHB,TAB10,LOG,LOGOK,TWO,LAWOK,LADOK,
A  FILFLG,QUIT,CHANGE,FIX,YES,NEEDED,YEP,NOPE,LIMITED,LALOK
C...
COMMON /WDCOMC/ KEY, ARG, INLINE(25), DATE
C...
CHARACTER KEY*4, ARG*4, INLINE*1, DATE*14
C
C... DATA DICTIONARY FOR COMMON BLOCK /WNDCOM/
C...
C... Purpose: General Program Variables
C
C  VK(33,3)   : WORKING ARRAY FOR INPUT & COMPUTED VARIABLES
C  PMOD(15,15): FUEL MODEL PARAMETERS
C  NMOD(15,8) : FUEL MODEL NAMES
C  NEEDED(30) : LOGICAL, .T. IF VARIABLE IS NEEDED ON INPUT
C  YEP        : LOGICAL, .T.
C  W20        : 20 FOOT WIND SPEED
C  WAF        : WIND ADJUSTMENT FACTOR
C  VERSN      : PROGRAM VERSION NUMBER
C  NTWO       : 1ST OR 2ND OF TWO FUEL MODELS
C  MODL       : ACTIVE FUEL MODEL
C  WMID       : MIDFLAME WIND SPEED
C  LIMITED(25): .T. IF VARIABLE HAS BEEN LIMITED (CONSTRAINED)
C  LAWOK      : .T. IF OK TO ZOOM OR REPLAY RX TABLE
C  LADOK      : .T. IF OK TO ZOOM OR REPLAY DEAD FM TABLE
C  LALOK      : .T. IF OK TO ZOOM OR REPLAY LIVE FM TABLE
C  PSLOP      : PERCENT SLOPE
C  FM(5)      : FRACTIONAL MOISTURE CONTENT OF EACH FUEL COMPONENT
C  OK         : GENERIC LOGICAL
C  WORDY      : .T. WHEN WORDY MODE, ELSE TERSE
C  PAUSE      : .T. WHEN PAUSE IS ON, ELSE NOPAUSE
C  MOI(5)     : .T. IF FUEL MOISTURE FOR COMPONENT IS REQUIRED
C  TABHB      : .T. IF LIVE TABLE VALUE IS HERB, ELSE WOODY
C  TAB10      : .T. IF DEAD TABLE VALUE IS 10-H, ELSE 1-H
C  LOG        : .T. IF LOG IS ON
C  LOGOK      : .T. IF LOG FILE OPENED
C  NOPE       : GENERIC .T.
C  TWO        : .T. IF TWO MODEL APPROACH USED (NEVER)
C  FILFLG     : .T. IF FUEL MODEL FILE ATTACHED AND OPEN
C  CHANGE     : .T. IF INPUT MODULE ENTERED IN CHANGE MODE
C  FIX        : .T. IF FIX IS NEEDED ON INPUT
C  YES        : GENERIC LOGICAL
C  LM1        : POINTER TO FIRST FUEL MODEL
C  LM2        : POINTER TO SECOND FUEL MODEL
C  REL100     : RELATION BETWEEN 10 AND 100 HOUR FM

```

C IEXPOS : EXPOSURE OF FUELS TO WIND (1-5)
C ITFB : POINTER TO TABLE FIRE BEHAVIOR (0-7, 0 = NONE)
C QUIT : .T. WHEN QUIT IS ENTERED
C...
C... DATA DICTIONARY FOR COMMON BLOCK /WNDCOMC/
C...
C... Purpose: Character values for program keyword control
C...
C KEY : 4 CHARACTER REPRESENTATION OF ENTERED KEYWORD
C ARG : 4 CHARACTER REPRESENTATION OF ENTERED KEYWORD ARGUMENT
C INLINE : CHARACTER ARRAY WITH INPUT LINE
C DATE : DATE OF PROGRAM VERSION (VERSN)
C...

Name : PROGRAM RXWINDOW (MAIN)

Purpose : MAIN program for KEYWORD functions.

Called by : None

Calls : DCODE, INRX, LIRX, CARX, LAWIND, LADEAD, LALIVE, HLPWIN,
KEYO, KEYALL, STATUS, COMENT, KEYFLG, SETUNT

System Routines: READ, WRITE

Parameter List : None

Common Includes: WNDCOM, IOCOM,

Variables Set : None

Description : Upon startup, welcome banner is displayed. It then displays keyword prompt and reads input. Passes input to DCODE and acts on returned value of KEY and ARG. All calls from MAIN return to MAIN. If KEY is LAST it calls appropriate routine based on ARG value. If KEY is QUIT, it makes sure you want to quit before terminating the RXWINDOW session.

Name : Subroutine ASK2

Purpose : Asks one line question with or without preceding line skip

Called by : INRX, READIT, READQ

Calls : None

System Routines: WRITE

Parameter List : LEN - length of prompt string
IQU - prompt string
SKIP - line skip flag

Common Includes: None

Variables Set : None

Description : Written by Andrews, modified by Chase and Bradshaw.

Name : BLOCK DATA

Purpose : Initialize common blocks

Called by : Program Initiation

Calls : None

System Routines: None

Parameter List : None

Common Includes: WNDCOM, IOCOM, SPCOM, UNITS

Variables Set : All COMMON constant values

Description : Initializes:

1. Logical Units in /IOCOM/
2. Standard Fuel Models in /WNDCOM/ and /SPCOM/
3. /UNITS/ and /UNTCHR/ constants
4. Sets program startup default values:

Prescription - Rate of Spread from 10 to 20 ch/h

Site Conditions - Fuel Model = 12
 WAF = .4 (exposed)
 Slope = 40%
 100-h = 10-h
 Tree Spc = 1
 Tree dbh = 8 in
 Tree hgt = 100 ft
 Ratio = 70%

Pre Set Limits - 1-h = 1 to 25 (none activated)
 10-h = 1 to 25
 100-h = 1 to 40
 Herb = 30 to 300
 Woody = 30 to 300
 20-ft wind = 0 to 30
 midflame wind = 0 to 25
 wind direction = uphill to downhill
 spread direction = head to back

5. Sets startup output table configurations

Prescription Table Fire Behavior Value: None
 Dead Moisture Table Value: 10-h tlfm
 Live Moisture Table Value: Herbaceous moisture

Name : SUBROUTINE CARX

Purpose : Determine Prescription Window

Called by : MAIN

Calls : SETMOI, SETUP, QSPREAD, WAITE, SHOWRX, CONTRL, READYN, CAMOIS

System Routines: NINT(), FLOAT()

Parameter List : None

Common Includes: WNDCOM, IOCOM, UNITS, QSPREAD, DISPLAY

Variables Set : INRX() - Prescription array and associated table values

Description :

1. Initialize table variables and INRX()
2. Determine weighted dead boundaries from 1 and 10 hour constraints and fuel model parameters.
3. Iterate for weighted dead fm within constraints...
 Iterate for weighted live fm (if needed) within constraints...
 Iterate for 20 foot wind within constraints...
 Test prescription (TESTRX, WINNER)
 Next wind
 Next live fm
 Next dead fm
4. If none in prescription, note and return
5. ENTRY POINT LAWIND: Set start, stop, and step values
6. Iterate for spread direction
 Show Table (SHOWRX)
 Determine Next Move (CONTL)
 Next direction
7. Call moisture module (CAMOIS) if needed
8. Return to MAIN

Name : SUBROUTINE CAMOIS

Purpose : Compute and display moisture tables

Called by : CARX

Calls : SETMOI, ZERO, QSPREAD, SHOWMD, CONTRL, SHOWML

System Routines: NINT, FLOAT, MINO, MAXO

Parameter List : None

Common Includes: WNDCOM, QSPREAD, IOCOM, UNITS, DISPLAY

Variables Set : MSD()

Description : 1. Initialize MSD()
2. Fill MSD() for dead moistures
3. Display table (SHOWMD)
4. Determine next move (CONTRL)
5. Return to CARX if QUIT requested in CONTRL.
6. If weighted live table not needed, return to CARX
7. Fill MSD() for live moistures
8. Display table (SHOWML)
9. Determine next move (CONTRL)
10. Return to CARX upon return from CONTRL.

Name : SUBROUTINE CHECK
 Purpose : Check input for valid form and values
 Called by : READIT, READQ, WHAT,
 Calls : READYN, CVRTU
 System Routines: NINT, MINO, FLOAT, IFIX, IABS
 Parameter List : V - 3 element array to hold start, step, step
 R1 - minimum allowed input value
 R2 - maximum allowed input value
 RNGOK - Range allowed flag
 IGR - Integer allowed flag
 NVAR - Pointer to VK() array for returned range values
 NVAR > 0, return input in VK() array, convert units
 using NVAR to point to CFAC() element.
 NVAR = 0, return input in VALUE, no units conversion
 NVAR < 0, return input in VALUE, convert units using
 IABS(NVAR) to point to CFAC() element.
 VALUE - scaler to put single value
 VK - array to put range values
 OK - flag if input is ok (.t.)
 Common Includes: UNITS
 Variables Set : VALUE or VK()
 Description : Written by Andrews, documented by Andrews and Bradshaw
 (1985). Modified by Bradshaw (1987) for switchable units.
 Modified again by Bradshaw for RXWINDOW to allow parameter
 VALUE to contain the maximum number of steps in a range input
 sequence (default is 7).

Name : SUBROUTINE CHKMOD

Purpose : Checks fuel model input

Called by : MAIN, INFUEL

Calls : READYN, READIT, MFLAG

System Routines: READ, WRITE

Parameter List : None

Common Includes: WNDCOM, IOCOM

Variables Set : LM1, LM2

Description : Documented in Andrews and Bradshaw (1985).

Name : SUBROUTINE COMENT

Purpose : Solicits and writes user comment to log file.

Called by : MAIN, CONTRL

Calls : None

System Routines: READ, WRITE

Parameter List : None

Common Includes: WNDCOM, IOCOM

Variables Set : TELL

Description : Written by Carolyn Chase. Solicits comment and writes it to log file until ** is entered. Modified by Bradshaw for RXWINDOW to ensure log functions are active and puts commenting instructions in COMENT instead of calling routines.

Name : SUBROUTINE CONTRL

Purpose : Provides all MAIN program KEYWORD functions, plus REPLAY and ZOOM from CARX and CAMOIS.

Called by : CARX, CAMOIS

Calls : DCODE, LIRX, ZOOMRX, ZOOMMD, ZOOMML, REPLRX, REPLMD, REPLML, HLPWIN, KEYO, KEYALL, STATUS, COMENT, KEYFLG, SETUNT

System Routines: READ, WRITE

Parameter List : FROM - Calling Module name (CHARACTER*4)
ITYPE - Fire spread direction code (passed to ZOOMRX & REPLRX)

Common Includes: WNDCOM, IOCOM

Variables Set : None

Description : Display keyword prompt and reads input. Passed input to DCODE and acts on returned value. If keyword is ZOOM or REPLAY it calls appropriate routine based on passed value FROM.

Name : SUBROUTINE CUSTOM

Purpose : Opens custom fuel model file and loads model from it. User may list models and model parameters.

Called by : INFUEL

Calls : READYN, WAITE, READIT, CVRTU

System Routines: READ, WRITE

Parameter List : None

Common Includes: UNITS, WNDCOM, IOCOM, KEEP

Variables Set : FILELG, PM (model parameters), NAME

Description : Written by Andrews and documented in Andrews and Bradshaw (1985). Modified by Bradshaw (1987) for switchable units, and further modified by Bradshaw for RXWINDOW to combine the functions of the CUSTOM keyword in FIRE1 and FIRE2 automatically into fuel input sequence in INFUEL for RXWINDOW when a model greater than 13 is requested.

Name : REAL FUNCTION CVRTU

Purpose : Converts values between unit systems

Called by : SETRNG, CHECK, CUSTOM, SCORLIM, LISTIT, SHOWRX

Calls : NONE

System Routines: ANINT, TAN, ATAN

Parameter List : VALUE - Value for conversion
CF - Conversion factor
CONV - Conversion flag
TOB - To base or not to base flag

Common Includes: UNITS

Variables Set : None

Description : Check if conversion needed. If so, check for direction of conversion. Then checks for special conversion functions (temperature and slope). Returns converted value as function value.

Name : SUBROUTINE DECODE

Purpose : Decodes input line into keyword and argument and converts to upper case.

Called by : MAIN, CONTRL

Calls : None

System Routines: ICHAR

Parameter List : LINE - Input line
KEY - Returned upper case keyword
ARG - Returned upper case argument following keyword

Common Includes: None

Variables Set : KEY, ARG

Description : Breaks LINE based on white space into KEY and ARG, converts to upper case and returns.

Name : SUBROUTINE FBTEST

Purpose : Tests computed fire behavior against prescription constraints.

Called by : TESTRX

Calls : None

System Routines: NINT

Parameter List : IFB - Fire behavior variable pointer
VK - working array (contains behavior constraints)
LOW - Low fire behavior flag
HIGH - High fire behavior flag
V1 - Computed Rate of spread
V2 - Computed heat per unit area
V3 - Computed flame length
V4 - Computed fireline intensity
V5 - Computed reaction intensity
V6 - Computed scorch height

Common Includes: None

Variables Set : LOW - True if fire behavior is low
HIGH - True if fire behavior is high

Description : Tests constrained prescriptions values against computed values. Is called selectively from TESTRX for constrained prescription parameters only.

Name : SUBROUTINE FLANK

Purpose : Computes fire behavior in a specific direction (SDIR)

Called by : TESTRX

Calls : None

System Routines: SQRT, ABS, COS

Parameter List : XDIR - Direction of maximum rate of spread
XRATE - Spread rate in SDIR (spread direction)
XBYRAM - Fireline intensity in SDIR
XFLAME - Flame length in SDIR
XSCOR77 - Scorch height in SDIR

Common Includes: WNDCOM, QSPREAD

Variables Set : XRATE, XBYRAM, XFLAME, XSCOR77

Description : Programed by Andrews, documented in Andrews and Bradshaw (1985).

Name : SUBROUTINE HLPWIN

Purpose : Provides help to user based on argument entered with keyword
HELP.

Called by : MAIN, CONTRL

Calls : None

System Routines: READ, WRITE

Parameter List : HLPARG - Help argument
FROM - Where help was called from.

Common Includes: WNDKOM, IOCOM

Variables Set : None

Description : Based on FROM and HLPARG, performs formatted writes.

Name : INTEGER FUNCTION IEOS

Purpose : Determines end of string position (last non-blank) in a character string.

Called by : READIT, READQ, INRX, LISTIT, SHOWRX, SHOWMD, SHOWML

Calls : None

System Routines: LEN

Parameter List : STRING - Variable length character string

Common Includes: None

Variables Set : IEOS - eos position

Description : Starts at end of string and steps back until non-blank is found. Used extensively for concatenations and printing unit labels.

Name : SUBROUTINE INFUEL

Purpose : Input fuel model

Called by : INRX

Calls : WHAT, CHKMOD, READYN, CUSTOM, READIT

System Routines: WRITE

Parameter List : None

Common Includes: WNDCOM, IOCOM

Variables Set : VK(8,1), VK(8,2), VK(8,3), MODL, TWO,

Description : 1. Prompts for fuel model number (1-99).
If quit entered, return.
2. If 1-13, set variables & return.
3. If > 13, ask if a custom model wanted.
If yes, call CUSTOM; else goto 1

Name : SUBROUTINE INRX

Purpose : Solicit, check, and store input parameters in VK()

Called by : MAIN

Calls : READQ, IEOS, ASK2, READYN, READIT, INPUT, WHAT, SETWAF,
SCORLIM, CVRTU, SETUP, QSPREAD, MORT

System Routines: WRITE

Parameter List : None

Common Includes: WNDCOM, IOCOM, UNITS, QSPREAD

Variables Set : LIMITED(), NEEDED(), VK()

Description : INRX is comprised of five basic sections:

1. Fire behavior constraint inputs (lines 1-7).
2. Site inputs (8-15).
3. Pre-set environmental constraint inputs (16-23).
4. Output configuration specifications (24-26).
5. Input checking

Each of the first four sections are do loop constructs with variable loop boundaries. The loop boundaries are set based on whether the entire input sequence is being executed (keyword INPUT), or one input is being addressed (keyword CHANGE).

Before exiting from INRX all inputs values are checked against potential conflicting parameters and the user is asked to resolve conflicts. Specifically:

1. At least one fire behavior variable must be constrained.
2. If both flame length and fireline intensity are constrained, one must be dropped.
3. If both heat per unit area and reaction intensity are constrained, one must be dropped.
4. If both mortality and scorch height are constrained, one must be dropped.
5. If both 20 foot and midflame wind are constrained, one must be dropped.

Name : SUBROUTINE KEYO

Purpose : Lists keywords for operational section of RXWINDOW.

Called by : MAIN, HLPWIN, CONTRL

Calls : None

System Routines: WRITE

Parameter List : None

Common Includes: IOCOM

Variables Set : None

Description : Writes functions of INPUT, LIST, CHANGE, RUN, QUIT keywords.

Name : SUBROUTINE KEYALL

Purpose : Lists keywords for environment of RXWINDOW.

Called by : MAIN, HLPWIN, CONTRL

Calls : None

System Routines: WRITE

Parameter List : None

Common Includes: IOCOM

Variables Set : None

Description : Writes functions of HELP, KEY, TERSE, WORDY, PAUSE, NOPAUSE,
LOG, NOLOG, STATUS, ENGLISH, METRIC, PERCENT, DEGREES,
COMMENT, ZOOM, REPLAY

Name : SUBROUTINE KEYFLG

Purpose : Check for keywords that set or clear flags WORDY, PAUSE, or LOG.

Called by : MAIN, CONTRL

Calls : LOGIT

System Routines: WRITE

Parameter List : None

Common Includes: IOCOM, WNDKOM

Variables Set : WORDY, PAUSE,

Description : Checks KEY against value of TERSE, WORDY, PAUSE, NOPAUSE, LOG, NOLOG and sets flag or calls LOGIT.

Name : SUBROUTINE LIFUEL

Purpose : List fuel model number, name and exposure to wind.

Called by : LIRX

Calls : None

System Routines: INDEX

Parameter List : LUOUT - Logical unit for output

Common Includes: WNDCOM, IOCOM

Variables Set : None

Description : Writes to LUOUT the fuel model line number (8), the model number, name and exposure of fuel to wind.

Name : SUBROUTINE LIRX

Purpose : Lists all RXWINDOW input to console or log file.

Called by : MAIN, CONTRL

Calls : LISTIT, WAITE, LIFUEL, IEOS

System Routines: WRITE, NINT

Parameter List : None

Common Includes: WNDCOM, IOCOM, UNITS

Variables Set : LUOUT

Description : Performs list operation once for console. If LOG is on,
repeats for log file.

Name : SUBROUTINE LISTIT

Purpose : General purpose listing of each input variable.

Called by : LIRX

Calls : CVRTU, IEOS

System Routines: IFIX, FLOAT, INDEX, LEN, NINT

Parameter List : NV - Pointer value of variable being listed
L - Line number of variable being listed
VK2 - Copy of working array VK
LUOUT - Logical unit to direct listing
IQU - String to print as variable label

Common Includes: WNDCOM, IOCOM

Variables Set : None

Description : Based on input values it prints the line number, variable label, and single or range of values. Format is based on size of value and variable type. Unit conversion are made if required.

Name : SUBROUTINE LOGIT

Purpose : Opens log file on initial LOG request and sets log flag.

Called by : KEYFLG

Calls : IEOS

System Routines: WRITE, READ, OPEN

Parameter List : ITSOPN - Log file flag
ITSON - Logging function flag

Common Includes: IOCOM

Variables Set : ITSOPN, ITSON, LOGFIL

Description : Opens log file on initial LOG request and sets ITSOPN and ITSON to true. Also prints banner. On NOLOG, it sets ITSON to false. Routine written by Roger Bradshaw, 1985. Modified by Larry Bradshaw to allow user make choice if a log file is requested that already exists. User may append to existing log file, change new log file name, or overwrite the existing log file with the new one.

Name : SUBROUTINE MFLAG

Purpose : Sets MOI and NEEDED flags base on fuel model.

Called by : CHKMOD

Calls : None

System Routines: None

Parameter List : None

Common Includes: WNDCOM, UNITS, IOCOM

Variables Set : MOI(), NEEDED()

Description : For each fuel component of a fuel model, MOI() and NEEDED() are set to true if the fuel load is greater than 0.

Name : SUBROUTINE MORT

Purpose : Compute probability of mortality base on stand descriptions
and scorch height based on model by Ryan and Reinhardt.

Called by : INRX, TESTRX

Calls : None

System Routines: MAX, EXP, MIN

Parameter List : ISPC - Tree species code
DBH - Tree dbh, in
SCHK - Scorch height, ft
RATIO - Crown ratio, percent
HEIGHT - Tree height, ft
PMORT - Probability of mortality
BARKT - Bark thickness, in
CVS - Crown volume scorched, %

Common Includes: None

Variables Set : PMORT

Description : Base on model by Ryan and Reinhardt computes PMORT based on
input parameters.

Name : SUBROUTINE QSPREAD

Purpose : Compute fuel bed, moisture, wind, slope, and fire behavior parameter.

Called by : TESTRX, INRX, CARX, CAMOIS

Calls : None

System Routines: EXP

Parameter List : ENTER - Subroutine entry point
LEAVE - Subroutine exit point

Common Includes: QSPREAD

Variables Set : Most of QSPREAD common

Description : Written by Cd. Bevins, 1987. Based on subroutine SPREAD from FIRE1 (Andrews, 1986). It is an optimized routine that permits partial calculation of intermediates, or computation of final values recomputing only intermediates that have changed since the last call. Was required for the interactive nature of the RXWINDOW design.

Name : SUBROUTINE READIT

Purpose : Ask a one line question, get and check the answer.

Called by : CHKMOD, INRX, CUSTOM, SETWAF, ZOOMRX, ZOOMMD, ZOOMML

Calls : IEOS, SETRNG, ASK2, VALUE, CHECK

System Routines: INDEX, IABS

Parameter List :
 IQU - String that contains the question
 R1 - Allowed lower limit
 R2 - Allowed upper limit
 RNGOK- Range allowed flag
 IGR - Integer allowed flag
 NVAR - Pointer to VK() position to put range results
 VALUE- Variable to hold single input result
 VK - Copy of working array

Common Includes: UNITS

Variables Set : VK, VALUE, OK

Description :
 1. Builds STR from IQU and UNTLBL() as required.
 2. Asks the question by passing STR to ASK2
 3. Solicits the answer in VALUES.
 4. Checks the answers. If ok, return, else go to 2.

Routine was written by Andrews and documented in Andrews and Bradshaw (1985). Modified by Bradshaw (1987) for switchable units.

Name : SUBROUTINE READQ

Purpose : Ask a one line question, get and check the answer, QUIT ok.

Called by : INRX

Calls : IEOS, SETRNG, ASK2, VALUE, CHECK

System Routines: INDEX, IABS

Parameter List :
 IQU - String that contains the question
 R1 - Allowed lower limit
 R2 - Allowed upper limit
 RNGOK- Range allowed flag
 IGR - Integer allowed flag
 NVAR - Pointer to VK() position to put range results
 VALUE- Variable to hold single input result
 VK - Copy of working array
 QUIT - Quit flag

Common Includes: UNITS

Variables Set : VK, VALUE, OK, QUIT

Description :
 1. Builds STR from IQU and UNTLBL() as required.
 2. Asks the question by passing STR to ASK2
 3. Solicits the answer in VALUES.
 4. Checks the answers. If ok, return, else go to 2.
 If input was 'quit', QUIT set to true.

Routine was written by Andrews and documented in Andrews and Bradshaw (1985). Modified by Bradshaw (1987) for switchable units.

Name : SUBROUTINE READYN

Purpose : Gets a yes or no answer

Called by : Many

Calls : None

System Routines: READ

Parameter List : YES

Common Includes: None

Variables Set : YES

Description : Read answer to a yes/ no question expecting Y, y, N, or n.

Name : SUBROUTINE REPLMD

Purpose : Replays last dead moisture table.

Called by : CNTRL

Calls : SHOWMD

System Routines: None

Parameter List : None

Common Includes: DISPLAY

Variables Set : None

Description : Calls SHOWMD with last values used in call to SHOWMD

Name : SUBROUTINE REPLML

Purpose : Replays last live moisture table.

Called by : CNTRL

Calls : SHOWML

System Routines: None

Parameter List : None

Common Includes: DISPLAY

Variables Set : None

Description : Calls SHOWML with last values used in call to SHOWML

Name : SUBROUTINE REPLRX

Purpose : Replays last prescription table.

Called by : CNTRL

Calls : SHOWRX

System Routines: None

Parameter List : ITYPE - Fire type code (head, flank, back)

Common Includes: DISPLAY

Variables Set : None

Description : Calls SHOWRX with last values used in call to SHOWRX

Name : REAL FUNCTION SCORLIM

Purpose : Returns maximum scorch height for a give level of mortality.

Called by : INRX

Calls : None

System Routines: LOG, MAX,, MIN, ABS

Parameter List : ISPC - Tree species code
 DBH - Tree dbh, in
 RATIO - Crown ratio, %
 HEIGHT - Tree height, ft
 PMORT - Specified mortality level

Common Includes: None

Variables Set : SCORLIM

Description : Based on reversal of equations by Ryan and Rienhardt,
 it compute the maximum allowable percent crown volume
 scorched and then determines the scorch height to achieve
 that level. Returns scorch height in feet.

Name : SUBROUTINE SETRNG

Purpose : Builds string RNGSTR with proper ranges.

Called by : READIT, READQ

Calls : CVRTU

System Routines: WRITE, FLOAT, NINT

Parameter List : RNGLOW - Active units lower range limit
RLOW - Base units lower range limit
RNGHIG - Active units upper range limit
RHIG - Base units upper range limit
IVAR - Pointer to active variable

Common Includes: UNITS

Variables Set : RNGSTR

Description : Builds RNGSTR with active unit values using RNGSTR as an internal file. RNGSTR contains the valid range for an input variable in the active units system. RNGSTR is then concatenated onto the prompt string in READIT or READQ.

Name : LOGICAL FUNCTION SETUNT

Purpose : Controls active unit selection and lists variables and units.

Called by : MAIN, CONTRL

Calls : None

System Routines: WRITE

Parameter List : None

Common Includes: UNITS, IOCOM, WND COM

Variables Set : SETUNT

Description : Tests a keyword entry for:

1. UNIT - If so, prompt for further UNIT keywords:
 - LIST - List all variable types and associated units
 - QUIT - Return to calling unit
 - ENGL, METR, PERC, DEGR (see below)
 - Loop to 1 until QUIT is entered.
2. ENGL - If so, set base flags on, return SETUNT= true.
3. METR - If so, set metric flags on, return SETUNT= true.
4. PERC - If so, set percent flags on, return SETUNT= true.
5. DEGR - If so, set degree flags on, return SETUNT= true.

If keyword is none of 1 to 5, return SETUNT= false.

Name : SUBROUTINE SETUP

Purpose : Sets up variables needed before call to QSPREAD

Called by : MAIN, INRX, CARX

Calls : None

System Routines: SAVE

Parameter List : None

Common Includes: WNDCOM, SPCOM, QSPREAD

Variables Set : FUELLOAD(), FUELHEAT(), FUELSAVR(), FUELDEPTH(), FUELMXT(),
NFC(), FUELSTOT(), FUELSEFF(), FUELDENS(),
WAF, WIND, SLOPE, FUELMOIS()

Description : Transfers values from PMOD(), and SPCOM common block.

Contains ENTRY SETMOI which only updates FUELLOAD(), WIND,
SLOPE, and FUELMOIS().

Name : REAL FUNCTION SETWAF

Purpose : Reads wind reduction factor after question has been asked.

Called by : INFUEL

Calls : READIT

System Routines: WRITE, NINT

Parameter List : ETW - Exposure to wind code

Common Includes: IOCOM, WNDKOM

Variables Set : SETWAF

Description : Based on value of ETW, sets SETWAF. If ETW = 5, SETWAF is directly input.

Name : SUBROUTINE SHOWMD

Purpose : Generates weighted dead moisture table.

Called by : CAMOIS, REPLMD, ZOOMMD

Calls : IEOS, WAITE

System Routines: WRITE

Parameter List : IDSTART - Weighted fm start value
 IDSTOP - Weighted fm stop value
 IDSTEP - Weighted fm step size
 I1STRT - 1-hr fm start value
 I1STOP - 1-hr fm stop value
 I1STEP - 1-hr fm step size
 NUM - Number of columns in table

Common Includes: WNDCOM, QSPREAD, IOCOM, UNITS, DISPLAY

Variables set : LDSTART - Weighted fm start value in last table
 LDSTOP - Weighted fm stop value in last table
 LDSTEP - Weighted fm step size in last table
 L1STRT - 1-hr fm start value in last table
 L1STOP - 1-hr fm stop value in last table
 L1STEP - 1-hr fm step size in last table
 LNUM - Number of columns in last table

Description : SHOWMD displays the contents of the moisture array,
 MSD(I,J,K) where J and K depend on the output table format:

Row Value (I)	Column Value (J)	Table Value (K:1=min,2=max)
Weighted dead fm	1-h dead fm	10-h dead fm range
Weighted dead fm	10-h dead fm	1-h dead fm range

Each cell within the passed parameter constraints is printed and the passed values are stored in the L series of variables for REPLY from CONTRL or MAIN.

MSD(I,J,K) is declared as an INTEGER*2.

Name : SUBROUTINE SHOWML

Purpose : Generates weighted live moisture table.

Called by : CAMOIS, REPLML, ZOOMML

Calls : IEOS, WAITE

System Routines: WRITE

Parameter List :

- LSTRT - Weighted live fm start value
- LSTOP - Weighted live fm stop value
- LSTEP - Weighted live fm step size
- IWSTRT - Woody fm start value
- IWSTOP - Woody fm stop value
- IWSTEP - Woody fm step size
- NUM - Number of columns in table

Common Includes: WNDCOM, QSPREAD, IOCOM, UNITS, DISPLAY

Variables set :

- LLSTRT - Weighted fm start value in last table
- LLSTOP - Weighted fm stop value in last table
- LLSTEP - Weighted fm step size in last table
- LWSTRT - 1-hr fm start value in last table
- LWSTOP - 1-hr fm stop value in last table
- LWSTEP - 1-hr fm step size in last table
- LWNUM - Number of columns in last table

Description : SHOWML displays the contents of the moisture array,
MSD(I,J,K) where J and K depend on the output table format:

Row Value (I)	Column Value (J)	Table Value (K:1=min,2=max)
Weighted live fm	Woody fm	Herbaceous fm range
Weighted live fm	Herbaceous fm	Woody fm range

Each cell within the passed parameter constraints is printed and the passed values are stored in the L series of variables for REPLY from CONTRL or MAIN.

MSD(I,J,K) is declared as an INTEGER*2.

Name : SUBROUTINE SHOWRX

Purpose : Generates weighted prescription window table.

Called by : CARX, REPLRX, ZOOMRX

Calls : IEOS, WAITE

System Routines: WRITE

Parameter List : IBEGW - Beginning 20 foot wind value
 IENDW - Ending 20 foot wind value
 ISTEPW - 20 foot wind step size
 IBEGD - Beginning weighted dead fm value
 IENDD - Ending weighted dead fm value
 ISTEPPD - Weighted dead fm step size
 ITYPE - Fire table type (head, flank, back)

Common Includes: WNDCOM, IOCOM, UNITS, DISPLAY

Variables Set : LWMN - Beginning 20 foot wind value last table
 LWMX - Ending 20 foot wind value last table
 LWST - 20 foot wind step size last table
 LDMN - Beginning weighted dead fm value last table
 LDMX - Ending weighted dead fm value last table
 LDST - Weighted dead fm step size last table

Description : SHOWRX displays the contents of the prescription array,
 INRX(I,J,K,L) where:

I = 20 foot wind speed (Column Value)
 J = Weighted dead fm (Row Value)
 K = Spread direction (Separate table for head, flank, back)
 L = 1: Frequency count
 2: Minimum weighted live fm for this prescription
 3: Maximum weighted live fm for this prescription
 4: Beginning range of wind direction, this prescription
 5: Ending range of wind direction, this prescription
 6: Minimum air temperature for this prescription
 7: Maximum air temperature for this prescription
 8: Minimum table fire behavior for this prescription
 9: Maximum table fire behavior for this prescription

INRX(I,J,K,L) is declared as an INTEGER*2.

Each cell within the passed parameter constraints is printed and the passed values are stored in the L series of variables for REPLY from CONTRL or MAIN.

For each row (weighted dead fm) the number of information rows depends on the fuel model, fire behavior prescription, slope, and if a fire behavior value is being printed in the table.

Name : SUBROUTINE STATUS

Purpose : Displays current status of program modules and modes.

Called by : MAIN, CNTRL

Calls : None

System Routines: WRITE

Parameter List : FROM - Where STATUS is called from
PROMPT - Status of prompt mode
SCREEN - Status of pause mode
LOGON - Status of log mode
LOGOPE - Status of log file
FFILE - Status of fuel model file

Common Includes: IOCOM, UNITS, KEEP

Variables Set : None

Description : Prints status of:

- Active Module
- Prompt Mode
- Pause Mode
- Log File Name
- Log Status (On or Off)
- Fuel File Name
- Display Units
- Slope Units

Name : SUBROUTINE TESTRX

Purpose : Monitor fire behavior computation and test for prescription.

Called by : CARX

Calls : SETMOI, QSPREAD, VECTOR, FLANK, FBTEST, MORT, WINNER

System Routines: NINT, FLOAT

Parameter List : LO - True if fire behavior is below rx
 HIG - True if fire behavior is above rx
 START - ENTER argument for call to QSPREAD
 STOP - LEAVE argument for call to QSPREAD
 ID - Pointer to weighted dead fm for INRX() assignment
 IW - Pointer to 20 foot wind for INRX() assignment
 IL - Value of weighted live fm

Common Includes: WNDCOM, UNITS, QSPREAD, DISPLAY

Variables Set : LO, HIGH (Returned from call to FBTEST)

Description : TESTRX is iteratively called from CARX within the two (or three) level moisture/wind loop. START and STOP vary to describe the enter/leave points on QSPREAD. Upon each initial entry to TESTRX, fire behavior is computed for head fire, wind blowing uphill. Then, fire behavior is iteratively evaluated at all non-constrained combinations of wind direction and fire spread direction, and for each combination, fire behavior is tested against prescription constraints by the call to FBTEST. If the combination is in prescription, WINNER is called to update the INRX() array for the current window pane.

NOTE: TESTRX uses an array TFB(8) that is EQUIVALENCED to fire behavior values in the /SPREADCB/ COMMON block for passing to the WINNER routine. This may cause a compiler warning that /SPREADCB/ is larger than in previous useage. The message may be ignored.

Name : SUBROUTINE TRANGE

Purpose : Returns air temperature range for a given range of scorch ht.

Called by : TESTRX, WINNER

Calls : None

System Routines: None

Parameter List : SC77 - Scorch height for 77 degree day
SCLO - Low range of desired scorch
SCHI - High range of desired scorch
TLO - Low temperature for high scorch
THI - High temperature for low scorch

Common Includes: None

Variables Set : TLO, THI

Description : Uses ratio described by Albini (1978) to determine ranges.

Name : SUBROUTINE VALUES

Purpose : Read alpha string and decode into numeric values within input rules.

Called by : WHAT, READIT, READQ

Calls : None

System Routines: FLOAT

Parameter List : V - 3 element array holding start, stop, step
OK - Input OK flag
QOK - QUIT allowed flag
QUIT - QUIT flag

Common Includes: IOCOM

Variables Set : OK, QUIT

Description : Written by Pat Andrews for FIRE1. Decodes character by character an input string checking for valid input syntax. Modified for RXWINDOW to allow two values only (simple range with no step), and negative values for relationship between 10 and 100 hour fuel moistures.

Name : SUBROUTINE VECTOR

Purpose : Finds the direction of maximum rate of spread given wind speed, wind direction, and slope.

Called by : TESTRX

Calls : None

System Routines: COS, SIN, SQRT, ASIN, ABS

Parameter List : XDIR - Direction of maximum spread
XRATE - Rate of spread in XDIR
XBYRAM - Fireline intensity in XDIR
XFLAME - Flame length in XDIR
XSCOR77- 77 degree scorch height in XDIR

Common Includes: WNDCOM, QSPREAD

Variables Set : XDIR, XRATE, XBYRAM, XFLAME, XSCOR77

Description : Written by Pat Andrews for FIRE1. Documented in Andrews and Bradshaw (1985).

Name : SUBROUTINE WAITE

Purpose : Pause display until enter key is pressed.

Called by : CUSTOM, LIRX, SHOWRX, SHOWMD, SHOWML, HLPWIN,

Calls : None

System Routines: WRITE, READ

Parameter List : None

Common Includes: WNDCOM, IOCOM

Variables Set : None

Description : If PAUSE is on, prompts for "enter to continue"

Name : SUBROUTINE WHAT
 Purpose : Read and check input values after a question already asked.
 Called by : INFUEL, INRX
 Calls : VALUES, CHECK
 System Routines: None
 Parameter List : R1 - Low range value
 R2 - Upper range value
 RNG - Range OK flag
 IGR - Integer OK flag
 NVAR - Pointer to VK() position to put range input
 VALUE - Scaler to put single input
 VK() - Working array
 OK - Input OK flag
 QUIT - QUIT flag
 QOK - QUIT Ok flag
 Common Includes: None
 Variables Set : VK(I,NVAR) or VALUE
 Description : Uses same logic as READIT and READQ except unit sensitive
 inputs are not allowed since question has already been asked
 and also flags whether QUIT is a valid input.

Name : SUBROUTINE WINNER

Purpose : Fills INRX() array with in-prescription information

Called by : TESTRX

Calls : None

System Routines: MINO, MAXO

Parameter List : ID - Weighted dead fuel moisture array position
IW - 20 foot wind array position
IL - Weighted live fuel moisture value
ISDS - Fire spread direction code
IWD - Wind direction code
IT - Scorch Temperature
IFBV - Table fire behavior variable value

Common Includes: DISPLAY

Variables Set : None

Description : Updates all table range values and fills all L components for the current INRX(IW,ID,ISDS,L) cell. (See SHOWRX for INRX definition.)

Name : SUBROUTINE ZERO

Purpose : Initialize MSD() array.

Called by : CAMOIS

Calls : None

System Routines: None

Parameter List : None

Common Includes: DISPLAY.INC

Variables Set : MSD()

Description : Minimum value initialized to 999
Maximum value initialized to -999

Name : SUBROUTINE ZOOMMD

Purpose : Zoom or pan current dead moisture array

Called by : CNTRL

Calls : READIT, SHOWMD

System Routines: NINT, FLOAT

Parameter List : None

Common Includes: WNDKOM, DISPLAY, IOCOM, UNITS

Variables Set : None

Description : Prompts user for corner bounds and step sizes of 1-h (or 10-hr) fm and weighted dead fm for zooming and calls SHOWMD to display requested array sections.

Name : SUBROUTINE ZOOMML

Purpose : Zoom or pan current live moisture array

Called by : CNTRL

Calls : READIT, SHOWML

System Routines: NINT, FLOAT

Parameter List : None

Common Includes: WNDCOM, DISPLAY, IOCOM, UNITS

Variables Set : None

Description : Prompts user for corner bounds and step sizes of live woody
(or herbaceous) fm and weighted live fm for zooming and calls
SHOWML to display requested array sections.

Name : SUBROUTINE ZOOMRX

Purpose : Zoom or pan current prescription array

Called by : CNTRL

Calls : READIT, SHOWRX

System Routines: NINT, FLOAT

Parameter List : ITYPE - Fire spread type code (1=head, 2=flank, 3=back)

Common Includes: WNDCOM, DISPLAY, IOCOM, UNITS

Variables Set : None

Description : Prompts user for corner bounds and step sizes of 20-foot wind speed and weighted dead fm for zooming and calls SHOWRX to display requested array sections for fire spread type ITYPE.

```
C...
C... COMMON BLOCK /DISPLAY/ FOR VARIABLE USED IN WINDOW
C... DISPLAY MODULES
C...
C...     PARAMETER (MXF=50,MXP=50,MX20=31,MXWD=30,MXS=3,MXI=9)
C...
C...     COMMON /DISPLAY/ LWMN,LWMX,LWST,LDMN,LDMX,LDST,ICOL(10,2),
1  LDSTRT,LDSTOP,LDSTEP,L1STRT,L1STOP,L1STEP,L1NUM,
2  LLSTRT,LLSTOP,LLSTEP,LWSTRT,LWSTOP,LWSTEP,LWNUM,
3  LOWI,MXW,MAXWI,LOD,MXD,MXLV,MNLV,
4  MAXMIN(MX20),MAXMAX(MX20),MAXWND(MX20),
5  MSD(MXF,MXP,2),INRX(MX20,MXWD,MXS,MXI),NONEIN
C...
C...     INTEGER*2 MSD,INRX
C...     LOGICAL NONEIN
C...
C...     DATA DICTIONARY FOR COMMON BLOCK /DISPLAY/
C...
C  LWMN   : MINIMUM 20 FOOT WIND FOR CURRENT WHOLE TABLE
C  LWMX   : MAXIMUM 20 FOOT WIND FOR CURRENT WHOLE TABLE
C  LWST   : WIND STEP SIZE FOR CURRENT WHOLE TABLE
C  LDMN   : MINIMUM WEIGHTED DEAD FM FOR CURRENT WHOLE TABLE
C  LDMX   : MAXIMUM WEIGHTED DEAD FM FOR CURRENT WHOLE TABLE
C  LDST   : WEIGHTED DEAD FM STEP SIZE FOR CURRENT WHOLE TABLE
C  ICOL() : ROW OUTPUT VALUE RANGES FOR 10 COLUMNS (1=MIN,2=MAX)
C
C  LDSTRT : LAST DISPLAY TABLE'S WEIGHTED DEAD FM START VALUE
C  LDSTOP : LAST DISPLAY TABLE'S WEIGHTED DEAD FM STOP VALUE
C  LDSTEP : LAST DISPLAY TABLE'S WEIGHTED DEAD FM STEP VALUE
C
C  L1STRT : LAST DISPLAY TABLE'S 1-HR DEAD FM START VALUE
C  L1STOP : LAST DISPLAY TABLE'S 1-HR DEAD FM STOP VALUE
C  L1STEP : LAST DISPLAY TABLE'S 1-HR DEAD FM STEP VALUE
C  L1NUM  : NUMBER OF STEPS ALLOWED IN TABLE
C
C  LLSTRT : LAST DISPLAY TABLE'S WEIGHTED LIVE FM START VALUE
C  LLSTOP : LAST DISPLAY TABLE'S WEIGHTED LIVE FM STOP VALUE
C  LLSTEP : LAST DISPLAY TABLE'S WEIGHTED LIVE FM STEP VALUE
C
C  LWSTRT : LAST DISPLAY TABLE'S WIND START VALUE
C  LWSTOP : LAST DISPLAY TABLE'S WIND STOP VALUE
C  LWSTEP : LAST DISPLAY TABLE'S WIND STEP VALUE
C  LWNUM  : NUMBER OF STEPS ALLOWED IN TABLE
C
C  LOD    : MINIMUM DEAD FUEL MOISTURE IN PRESCRIPTION
C  MXD    : MAXIMUM DEAD FUEL MOISTURE IN PRESCRIPTION
C  LOWI   : MINIMUM 20 FOOT WIND IN PRESCRIPTION
C  MXW    : MAXIMUM 20 FOOT WIND IN PRESCRIPTION
C  MXLV   : MAXIMUM WEIGHTED LIVE FUEL MOISTURE IN OUTPUT TABLE
C  MNLV   : MINIMUM WEIGHTED LIVE FUEL MOISTURE IN OUTPUT TABLE
C
C  MAXMIN(): HIGHEST MINIMUM LIVE FUEL MOIST FOR A GIVEN 20 FOOT WIND
C  MAXMAX(): HIGHEST MAXIMUM LIVE FUEL MOIST FOR A GIVEN 20 FOOT WIND
C  MAXWND(): MAXIMUM 20 FOOT WIND FOR ENTIRE PRESCRIPTION RANGE
```

C MSD(I,J,K): FOR A GIVEN WEIGHTED MOISTURE CONTENT (I), AND 1-HOUR
C FM (J), K=1 IS MINIMUM ALLOWED 10-HOUR FM AND K=2 IS
C MAXIMUM ALLOWED 10-HOUR FM. IF THE USER HAS SELECTED
C 10-HOUR ACROSS THE TOP OF THE TABLE, THEN THE VALUES
C IN THE (J) AND (K) SLOTS ARE REVERSED.

C INRX(I,J,K,L) : I = WIND SPEED
C J = WEIGHTED DEAD FUEL MOISTURE
C K = FIRE SPREAD DIRECTION (HEAD,FLANK,REAR)
C L = POINTER VALUES:
C 1 = KOUNTER OF NUMBER IN RX
C 2 = MIN WEIGHTED LIVE FM RX
C 3 = MAX WEIGHTED LIVE FM RX
C 4 = BEGIN WIND DIRECTION IN RX
C 5 = END WIND DIRECTION IN RX
C 6 = MIN TEMP FOR SCORCH IN RX
C 7 = MAX TEMP FOR SCORCH IN RX
C 8 = MIN TABLE FIRE BEHAVIOR VALUE IN CELL
C 9 = MAX TABLE FIRE BEHAVIOR VALUE IN CELL

C NONEIN : .TRUE. = CURRENT RX HAS NO SOLUTION\

```
C
C.... COMMON BLOCK FOR ALL IO NEEDS
COMMON /INOUT/ LU4,LU5,LU6,LU7
COMMON /INOUTC/ LOGFIL,FILNAM
CHARACTER LOGFIL*12,FILNAM*12
C...
C... DATA DICTIONARY FOR /INOUT/
C
C    LU4 : LOGICAL UNIT FOR LOG FILE
C    LU5 : LOGICAL UNIT FOR CONSOLE WRITE
C    LU6 : LOGICAL UNIT FOR CONSOLE READ
C    LU7 : LOGICAL UNIT FOR CUSTOM FUEL MODEL FILE
C...
C... DATA DICTIONARY FOR /INOUTC/
C
C    LOGFIL : CHARACTER*12, NAME OF LOG FILE
C    FILNAM : CHARACTER*12, NAME OF FUEL FILE
C
```

```
C-----
C PARAMETERS
C-----
      INTEGER NCLASS,NCATEG,DEAD,LIVE,HR1,HR10,HR100
      PARAMETER (NCLASS=3,NCATEG=2,DEAD=1,LIVE=2,HR1=1,HR10=2,HR100=3)
C-----
C /SPREADCB/ VARIABLE TYPES
C-----
C...NUMBER FUEL CLASSES
      INTEGER NFC
C...FUEL INPUTS
      REAL FUELLOAD,FUELSAVR,FUELSTOT,FUELSEFF,FUELHEAT,FUELDENS,
      + FUELDEPTH,FUELMEXT
C...ENVIRONMENTAL INPUTS
      REAL FUELMOIS,SLOPE,SDIR,WIND,WDIR,WMAX
C... INTERMEDIATE VARIABLES
      REAL FUELAREA,FUELG,FCATAREA,FCATLOAD,FCATSAVR,FCATETAS,FCATSEFF,
      + FCATHEAT,BETAOP,AA,GAMMA,GAMMAX,C,E
C...DERIVED FUEL BED CONSTANTS
      REAL FUELF,FUELMWF,FCATF,PRI,FINED,WLIVE,SIGMA,BETA,RATIO,RHOB,
      + XI,KSLOPE,B,KWIND
C...DERIVED ENVIRONMENTAL INTERMEDIATES
      REAL QIG,FCATMOIS,FCATETAM,FCATMEXT,WMFD,FDMOIS
C...DERIVED ENVIRONMENTAL CONSTANTS
      REAL RBQIG,XIR,PHIW,PHIS,PHIEW,EWIND,LWIND,WLIM
C...FIRE BEHAVIOR RESULTS
      REAL RATE0,RATE,HPUA,BYRAM,FLAME,SCOR77
C-----
C /SPREADCB/ OPTIMIZED SPREAD COMMON BLOCK
C      CONTAINS 2 INTEGER*4 AND 132 REALS*4 (536 BYTES)
C      AS DECLARED AND DEFINED ABOVE
C-----

      COMMON /SPREADCB/
      + NFC(NCATEG),
      + FUELLOAD(NCLASS,NCATEG),FUELSAVR(NCLASS,NCATEG),
      + FUELSTOT(NCLASS,NCATEG),FUELSEFF(NCLASS,NCATEG),
      + FUELHEAT(NCLASS,NCATEG),FUELDENS(NCLASS,NCATEG),
      + FUELDEPTH,FUELMEXT,
      + FUELMOIS(NCLASS,NCATEG),SLOPE,SDIR,WIND,WDIR,WMAX,
      + FUELAREA(NCLASS,NCATEG),FUELG(NCLASS,NCATEG),FCATAREA(NCATEG),
      + FCATLOAD(NCATEG),FCATSAVR(NCATEG),FCATETAS(NCATEG),
      + FCATSEFF(NCATEG),FCATHEAT(NCATEG),BETAOP,AA,GAMMA,GAMMAX,
      + FUELF(NCLASS,NCATEG),FUELMWF(NCLASS,NCATEG),FCATF(NCATEG),
      + PRI(NCATEG),FINED,WLIVE,SIGMA,BETA,RATIO,RHOB,XI,KSLOPE,
      + B,C,E,KWIND,
      + QIG(NCLASS,NCATEG),FCATMOIS(NCATEG),FCATETAM(NCATEG),
      + FCATMEXT(NCATEG),WMFD,FDMOIS,
      + RBQIG,XIR,PHIW,PHIS,PHIEW,EWIND,LWIND,WLIM,
      + RATE0,RATE,HPUA,BYRAM,FLAME,SCOR77
C...
```

C... COMMON BLOCK DEFINITION FOR /SPCOM/ - SPREAD VARIABLES

```
COMMON /SPCOM/ XLOAD,SIG,STOT,SEFF,HEAT,XMOIS,DENS,NCLAS
DIMENSION XLOAD(2,3),SIG(2,3),STOT(2,3),SEFF(2,3),HEAT(2,3),
- XMOIS(2,3),DENS(2,3),NCLAS(2)
```

C

C... DATA DICTIONARY FOR COMMON BLOCK /SPCOM/

C

C XLOAD : FUEL LOADS

C SIG : SURFACE AREA TO VOLUME RATIO

C STOT : TOTAL MINERAL CONTENT

C SEFF : EFFECTIVE MINERAL CONTENT

C HEAT : FUEL HEAT CONTENT

C XMOIS : FUEL EXTINCTION MOISTURE

C DENS : FUEL PARTICLE DENSITY

C NCLASS : # FUEL SIZE CLASS/COMPONENT

C...

```
C... COMMON BLOCK INCLUDE FILE (UNITS.COM) FOR UNIT CONVERSIONS AND
C... VARIABLE NAME DEFINITIONS
C...
C... /UNITS/ CONTAINS POINTERS, LOGICAL FLAGS, RANGE AND CONVERSION ARRAYS
C...
    LOGICAL METRIC, TOBASE, NOT2BA, PERCENT
    COMMON /UNITS/ IUNT, IBAS, IMET, ISLP, IPCT, IDEG, LONG, ISHT, LIML, LIMH,
    A RNG(2,33), CFAC(33), METRIC, TOBASE, NOT2BA, PERCENT,
    B IROS, IHUA, IFLE, IFLI, IRIN, ISCO, IMOR,
    C IFUL, IWAF, ISLO, IREL, ISPC, IDBH, IHGT, ICRN,
    D I1HR, I10H, IHR8, IWDY, IW20, IWMD, IWDD, ISDD,
    E IW20B, I1DW, I1MD, IWGHT, IWDYL, IWLX, ITMP,
    F IUN1, IUN2, NON
C...
C... /UNTCHR/ CONTAINS CHARACTER CONSTANTS
    CHARACTER UNTLBL*10, STR*70, RNGSTR*20, VARLBL*40
    COMMON /UNTCHR/ UNTLBL(2,2,33), VARLBL(30), STR, RNGSTR
C...
C... DATA DICTIONARY FOR /UNITS/
C... /UNITS/ CONSISTS PRIMARILY OF POINTERS TO ARRAY SLOTS FOR ALL
C... DISPLAYABLE VARIABLES
C    IUNT    : ACTIVE UNIT SET (1=BASE, 2=METRIC)
C    IBAS    : BASE UNIT POINTER
C    IMET    : METRIC UNIT POINTER
C    ISLP    : ACTIVE SLOPE UNITS (1=PERCENT, 2=DEGREES)
C    IPCT    : SLOPE IN PERCENT UNIT POINTER
C    IDEG    : SLOPE IN DEGREES UNIT POINTERS
C    LONG    : POINTER TO LONG UNITS LABELS
C    ISHT    : POINTER TO SHORT UNITS LABELS
C    LIML    : POINTER TO INPUT VALUE LOW RANGE VALUE
C    LIMH    : POINTER TO INPUT VALUE HIGH RANGE VALUE
C    RNG()   : ALLOWABLE INPUT RANGE FOR A GIVEN VARIABLE
C
C    CFAC()  : CONVERSION FACTOR FOR A GIVEN VARIABLE
C    METRIC  : METRIC UNITS WHEN .T.
C
C    TOBASE  : CONVERT INPUT TO BASE VALUE WHEN TRUE
C    NOT2BA  : DO NOT CONVERT INPUT TO BASE WHEN TRUE
C    PERCENT : SLOPE UNITS IN PERCENT WHEN TRUE
C
C    IROS    : POINTER TO RATE OF SPREAD
C    IHUA    : POINTER TO HEAT PER UNIT AREA
C    IFLE    : POINTER TO FLAME LENGTH
C    IFLI    : POINTER TO FIRELINE INTENSITY
C    IRIN    : POINTER TO REACTION INTENSITY
C    ISCO    : POINTER TO SCORCH HEIGHT
C    IMOR    : POINTER TO MORTALITY
C
C    IFUL    : POINTER TO FUEL MODEL
C    IWAF    : POINTER TO WIND ADJUSTMENT FACTOR
C    ISLO    : POINTER TO SLOPE
C    IREL    : POINTER TO 10 TO 100 HOUR RELATION
C    ISPC    : POINTER TO SPECIES
C    IDBH    : POINTER TO DIAMETER BREAST HEIGHT
```

```
C      IHGT      : POINTER TO TREE HEIGHT
C      ICRN      : POINTER TO CROWN RATIO

C      I1HR      : POINTER TO 1 HOUR MOISTURE
C      I10H      : POINTER TO 10 HOUR MOISTURE
C      IHR8      : POINTER TO HERBACEOUS MOISTURE
C      IWDY      : POINTER TO WOODY MOISTUR
C      IW20      : POINTER TO 20 FOOT WIND
C      IWMD      : POINTER TO MIDFLAME WIND
C      IWDD      : POINTER TO WIND DIRECTION
C      ISDD      : POINTER TO SPREAD DIRECTION

C      IW20B     : POINTER TO 20 FOOT FROM ZOOM ROUTINES
C      I1DW      : POINTER TO 1 HOUR DEAD FROM ZOOM ROUTINES
C      I1MD      : POINTER TO 1 HOUR DEAD FROM ZOOM ROUTINES
C      IWGHT     : POINTER TO WEIGHTED DEAD FROM ZOOM ROUTINES
C      IWDYL     : POINTER TO LIVE HERB MOISTURE FROM ZOOM ROUTINES
C      IWLVS     : POINTER TO WEIGHTED LIVE FROM ZOOM ROUTINES

C      ITMP      : POINTER TO AIR TEMPERATURE

C      IUN1      : POINTER TO UNUSED 1
C      IUN2      : POINTER TO UNUSED 2
C      NON       : POINTER TO NO UNITS

C... DATA DICTIONARY FOR COMMON BLOCK /UNTCHR/
C
C      UNTLBL(I,J,K) : UNIT LABELS — I = BASE/METRIC
C                                   J = LONG/SHORT
C                                   K = VARIABLE
C
C      VARLBL(K)     : VARIABLE LABEL
C      STR           : UTILITY CHARACTER STRING
C      RNGSTR        : STRING TO HOLD RANGE FOR PROMPTS WITH RANGES
C
```

C... COMMON BLOCK DEFINITION FOR /WINDCOM/

C...

```
COMMON /WINDCOM/ VK(33,3), PMOD(15,15), NMOD(15,8), NEEDED(30), YEP,  
A W20, WAF, VRSION, NTWO, MODL, WMID, LIMITED(25), LAWOK, LADOK, LALOK,  
B PSLOP, FM(5), OK, WORDY, PAUSE, MOI(5), TABHB, TAB10, LOG, LOGOK, NOPE,  
C TWO, FILFLG, CHANGE, FIX, YES, LM1, LM2, REL100, IEXPOS, ITFB, QUIT
```

C...

```
LOGICAL OK, WORDY, PAUSE, MOI, TABHB, TAB10, LOG, LOGOK, TWO, LAWOK, LADOK,  
A FILFLG, QUIT, CHANGE, FIX, YES, NEEDED, YEP, NOPE, LIMITED, LALOK
```

C...

```
COMMON /WINDCOMC/ KEY, ARG, INLINE(25), DATE
```

C...

```
CHARACTER KEY*4, ARG*4, INLINE*1, DATE*14
```

C

C... DATA DICTIONARY FOR COMMON BLOCK /WINDCOM/

C

```
C VK(33,3) : WORKING ARRAY FOR INPUT & COMPUTED VARIABLES  
C PMOD(15,15): FUEL MODEL PARAMETERS  
C NMOD(15,8) : FUEL MODEL NAMES  
C NEEDED(30) : LOGICAL, .T. IF VARIABLE IS NEEDED ON INPUT  
C YEP : LOGICAL, .T.  
  
C W20 : 20 FOOT WIND SPEED  
C WAF : WIND ADJUSTMENT FACTOR  
C VRSION : PROGRAM VERSION NUMBER  
C NTWO : 1ST OR 2ND OF TWO FUEL MODELS  
C MODL : ACTIVE FUEL MODEL  
C WMID : MIDFLAME WIND SPEED  
C LIMITED(25): .T. IF VARIABLE HAS BEEN LIMITED (CONSTRAINED)  
C LAWOK : .T. IF OK TO ZOOM OR REPLAY RX TABLE  
C LADOK : .T. IF OK TO ZOOM OR REPLAY DEAD FM TABLE  
C LALOK : .T. IF OK TO ZOOM OR REPLAY LIVE FM TABLE  
C PSLOP : PERCENT SLOPE  
C FM(5) : FRACTIONAL MOISTURE CONTENT OF EACH FUEL COMPONENT  
C OK : GENERIC LOGICAL  
C WORDY : .T. WHEN WORDY MODE, ELSE TERSE  
C PAUSE : .T. WHEN PAUSE IS ON, ELSE NOPAUSE  
C MOI(5) : .T. IF FUEL MOISTURE FOR COMPONENT IS REQUIRED  
C TABHB : .T. IF DEAD TABLE VALUE IS 10-H, ELSE 1-H  
C TAB10 : .T. IF LIVE TABLE VALUE IS HERB, ELSE WOODY  
C LOG : .T. IF LOG IS ON  
C LOGOK : .T. IF LOG FILE OPENED  
C NOPE : GENERIC .T.  
C TWO : .T. IF TWO MODEL APPROACH USED (NEVER)  
C FILFLG : .T. IF FUEL MODEL FILE ATTACHED AND OPEN  
C CHANGE : .T. IF INPUT MODULE ENTERED IN CHANGE MODE  
C FIX : .T. IF FIX IS NEEDED ON INPUT  
C YES : GENERIC LOGICAL  
C LM1 : POINTER TO FIRST FUEL MODEL  
C LM2 : POINTER TO SECOND FUEL MODEL  
C REL100 : RELATION BETWEEN 10 AND 100 HOUR FM  
C IEXPOS : EXPOSURE OF FUELS TO WIND (1-5)  
C ITFB : POINTER TO TABLE FIRE BEHAVIOR (0-7, 0 = NONE)  
C QUIT : .T. WHEN QUIT IS ENTERED
```

C
C... DATA DICTIONARY FOR /WINDCOMC/
C...
C... Purpose: Character values for program keyword control
C...
C KEY : 4 CHARACTER REPRESENTATION OF ENTERED KEYWORD
C ARG : 4 CHARACTER REPRESENTATION OF ENTERED KEYWORD ARGUMENT
C INLINE : CHARACTER ARRAY WITH INPUT LINE
C DATE : DATE OF PROGRAM VERSION VRSION
C...

PROGRAM RXWINDOW

```
C
C////PROGRAMMED BY LARRY BRADSHAW, SEM, 1986-88
C
C .... DRIVER PROGRAM
C      CALLS MODULES BASED ON KEYWORD INPUT
C
C      INCLUDE 'WND.COM.INC'
C      INCLUDE 'IO.COM.INC'
C
C      LOGICAL SETUNT
C
C      OPEN(UNIT=LU5,FILE='CON')
C      OPEN(UNIT=LU6,FILE='CON',CARRIAGE CONTROL='LIST')
C      VRSION = 2.8
C      DATE = 'JUNE 88 '
C
C .... WELCOME ....
C      WRITE (LU6,6000) VRSION,DATE
6000 FORMAT(// ' PRESCRIPTION WINDOW PROGRAM: VERSION ',F3.1,' -- ',A14,
-          // '   DEVELOPED BY THE FIRE BEHAVIOR RESEARCH WORK UNIT, '
-          // '   INTERMOUNTAIN FIRE SCIENCES LABORATORY, '
-          // '   IN COOPERATION WITH SYSTEMS FOR ENVIRONMENTAL MGMT. '
-          // '   MISSOULA, MONTANA' )
C .... INITIALIZE LOG FILE STATUS
C      LOGOK=.FALSE.
C      LOG=.FALSE.
C
C...THESE STATEMENTS SKIP OPENING INTRO CRT CONSOLE CHECK.
C      PAUSE = .TRUE.
C...      GO TO 1
CXXX      WRITE (LU6,6070)
CXXX 6070 FORMAT (//46H ARE YOU USING A TERMINAL WITH A SCREEN ? Y-N )
CXXX      CALL READYN(YES)
CXXX      IF (YES) THEN
C          WRITE (LU6,6080)
6080 FORMAT (// ' PAUSE OPTION SET. YOU WILL BE PROMPTED TO CONTINUE. ' )
CXXX      -          // ' WHEN YOU ARE READY TO CONTINUE AFTER THE PROMPT SYMB',
CXXX      -'OL IS PRINTED'
CXXX      -          // ' WITHOUT A QUESTION, PRESS THE CARRIAGE RETURN KEY. ' )
CXXX      PAUSE=.TRUE.
CXXX      ELSE
CXXX      PAUSE=.FALSE.
CXXX      ENDIF
C
C      1 CONTINUE
C
CXXX      CALL CHKMOD
CXXX      CALL SETUP
C
C      10 CONTINUE
C      WRITE (LU6,1010)
1010 FORMAT (/16H WINDOW KEYWORD? )
C      IF (WORDY) WRITE (LU6,1020)
```

```
1020 FORMAT(' ENTER:  INPUT,LIST,CHANGE,RUN,HELP,KEY'  
& '/'      WORDY,TERSE,PAUSE,NOPAUSE,COMMENT,ENGLISH,METRIC'  
& '/'      PERCENT,DEGREES,LOG,NOLOG,STATUS,QUIT')  
C  
  IF (LAWOK .AND. WORDY) THEN  
    IF (MOI(4).AND.MOI(5)) THEN  
      WRITE(LU6,"('      LAST WINDOW, LAST DEAD, LAST LIVE')")  
    ELSE  
      WRITE(LU6,"('      LAST WINDOW, LAST DEAD')")  
    ENDIF  
  ENDIF  
  
  READ (LU5,1110) INLINE  
1110 FORMAT (25A1)  
  CALL DCODE(INLINE,KEY,ARG)  
C  
C .... KEYWORD = INPUT  
  IF (KEY(1:1).EQ.'I') THEN  
    CHANGE = .FALSE.  
    CALL INRX  
    LAWOK = .FALSE.  
    GO TO 10  
  ENDIF  
C  
  IF (KEY(1:2).EQ.'LA') THEN  
    IF (LAWOK) THEN  
      IF (ARG .EQ. 'WIND') CALL LAWIND  
      IF (ARG .EQ. 'DEAD') CALL LADEAD  
      IF (ARG .EQ. 'LIVE') THEN  
        IF(MOI(4).AND.MOI(5)) THEN  
          CALL LALIVE  
        ELSE  
          WRITE(LU6,"('/ NO LIVE MOISTURE TABLE AVAILABLE...')")  
        ENDIF  
      ENDIF  
    ELSE  
      WRITE(LU6,"('/ LAST NOT ALLOWED UNTIL RUN IS ENTERED...')")  
    ENDIF  
    GOTO 10  
  ENDIF  
C...  
C... KEYWORD ZOOM (NOT ALLOWED HERE)  
  IF (KEY(1:1) .EQ. 'Z') THEN  
    WRITE(LU6, "('/ ZOOM ONLY ALLOWED FROM DISPLAY MODULE...')")  
    GOTO 10  
  ENDIF  
C... KEYWORD REPLAY (NOT ALLOWED HERE)  
  IF (KEY(1:2) .EQ. 'RE') THEN  
    WRITE(LU6, "('/ REPLAY ONLY ALLOWED FROM DISPLAY MODULE...')")  
    GOTO 10  
  ENDIF  
C... KEYWORD CONTINUE (NOT ALLOWED HERE)  
  IF (KEY(1:3) .EQ. 'CON') THEN
```

```
WRITE(LU6, "(/' CONTINUE ONLY ALLOWED FROM DISPLAY MODULE...')")
GOTO 10
ENDIF
```

```
C .... KEYWORD = LIST
IF (KEY(1:2).EQ.'LI') THEN
  CALL LIRX
  GO TO 10
ENDIF
```

```
C
IF (KEY(1:2).EQ.'CH') THEN
  CHANGE = .TRUE.
  CALL INRX
  LAWOK = .FALSE.
  GO TO 10
ENDIF
```

```
C
IF (KEY(1:2).EQ.'RU') THEN
  CALL CARX
  GO TO 10
ENDIF
```

```
C
IF (KEY(1:1).EQ.'H') THEN
  CALL HLPWIN(ARG, 'MAIN')
  GO TO 10
ENDIF
```

```
C
IF (KEY(1:1).EQ.'K') THEN
  CALL KEY0
  CALL KEYALL
  GO TO 10
ENDIF
```

```
C
IF (KEY(1:1).EQ.'S') THEN
  CALL STATUS('MAIN',WORDY,PAUSE,LOG,LOGOK,FILFLG)
  GOTO 10
ENDIF
```

```
C
IF (KEY(1:3).EQ.'COM') THEN
  CALL COMENT
  GO TO 10
ENDIF
```

```
C
C
IF (KEY(1:1).EQ.'Q') THEN
  WRITE (LU6,1250)
```

```
1250 FORMAT(// 'DO YOU R E A L L Y WANT TO TERMINATE THIS RUN? Y-N')
```

```
CALL READYN(YES)
IF (.NOT.YES) THEN
```

```
WRITE (LU6,1260)
```

```
1260 FORMAT (/10H OK..... /)
```

```
GO TO 10
```

```
ENDIF
```

```
GO TO 999
```

```
ENDIF
```

C

```
CALL KEYFLG
IF (OK) GO TO 10
```

```
C... CHECK FOR UNITS KEY WORDS
IF (SETUNT() ) GOTO 10
```

C

```
WRITE (LU6,1130) KEY
1130 FORMAT (23H UNRECOGNIZED KEYWORD—,A4)
GO TO 10
```

C

```
C .... TERMINATE
999 CONTINUE
```

```
IF (LOGOK) THEN
```

```
WRITE (LU6,1380) LOGFIL
```

```
1380 FORMAT (/ ' PART OF THIS RUN MAY HAVE BEEN LOGGED. '
```

```
-      / ' THE FILE NAME IS:  ',A12
```

```
-      / ' PRINT THE FILE NOW AND DELETE IT. ')
```

```
CLOSE (LU4)
```

```
ENDIF
```

C

```
CLOSE (LU7)
```

```
WRITE (LU6,2000)
```

```
2000 FORMAT (// ' THE WINDOW IS NOW CLOSED.  '/1X,30(1H*)//)
```

C

```
STOP
```

```
END
```

BLOCK DATA

C INCLUDE 'WDCOM.INC'

C INCLUDE 'IOCOM.INC'

C INCLUDE 'SPCOM.INC'

C INCLUDE 'UNITS.INC'

C... DATA LU4,LU5,LU6,LU7 /3,0,0,7/

DATA LU4,LU5,LU6,LU7 /3,5,10,7/

C LU4 - I/O LOGICAL UNIT - WRITE TO LOG FILE

C LU5 - I/O LOGICAL UNIT - READ FROM CONSOLE

C LU6 - I/O LOGICAL UNIT - WRITE TO CONSOLE

C LU7 - I/O LOGICAL UNIT - READ FROM FUEL MODEL FILE

C... ASSIGN INTERNAL VALUES TO VARIABLES IN COMMON BLOCKS

DATA SIG(1,2)/109./,SIG(1,3)/30./,NCLAS(1)/3/,NCLAS(2)/2/,

- ((STOT(I,J),I=1,2),J=1,3)/6*.0555/,

- ((SEFF(I,J),I=1,2),J=1,3)/6*.0100/,

- ((DENS(I,J),I=1,2),J=1,3)/6*32./

C... FUEL MODELS

DATA (PMOD(1,J),J=1,15)/.034,0.,0.,0.,0.,3500.,99.,99.,

- 1.0,8000.,.12,0.,.4,0.,0./,

- (PMOD(2,J),J=1,15)/.092,.046,.023,.023,0.,3000.,1500.,99.,

- 1.0,8000.,.15,0.,.4,0.,0./,

- (PMOD(3,J),J=1,15)/.138,0.,0.,0.,0.,1500.,99.,99.,

- 2.5,8000.,.25,0.,.4,0.,0./,

- (PMOD(4,J),J=1,15)/.230,.184,.092,0.,.230,2000.,99.,1500.,

- 6.0,8000.,.20,0.,.6,0.,0./,

- (PMOD(5,J),J=1,15)/.046,.023,0.,0.,.092,2000.,99.,1500.,

- 2.0,8000.,.20,0.,.4,0.,0./,

- (PMOD(6,J),J=1,15)/.069,.115,.092,0.,0.,1750.,99.,99.,

- 2.5,8000.,.25,0.,.4,0.,0./,

- (PMOD(7,J),J=1,15)/.052,.086,.069,0.,.017,1750.,99.,1500.,

- 2.5,8000.,.40,0.,.4,0.,0./,

- (PMOD(8,J),J=1,15)/.069,.046,.115,0.,0.,2000.,99.,99.,

- 0.2,8000.,.30,0.,.4,0.,0./

DATA (PMOD(9,J),J=1,15)/.134,.019,.007,0.,0.,2500.,99.,99.,

- 0.2,8000.,.25,0.,.4,0.,0./,

- (PMOD(10,J),J=1,15)/.138,.092,.230,0.,.092,2000.,99.,1500.,

- 1.0,8000.,.25,0.,.4,0.,0./,

- (PMOD(11,J),J=1,15)/.069,.207,.253,0.,0.,1500.,99.,99.,

- 1.0,8000.,.15,0.,.4,0.,0./,

- (PMOD(12,J),J=1,15)/.184,.644,.759,0.,0.,1500.,99.,99.,

- 2.3,8000.,.20,0.,.4,0.,0./,

- (PMOD(13,J),J=1,15)/.322,1.058,1.288,0.,0.,1500.,99.,99.,

- 3.0,8000.,.25,0.,.5,0.,0./

C... FUEL MODEL NAMES

```

DATA (NMOD(1,J),J=1,8)
- /'SHOR','T GR','ASS','1 F','T (3','0 CM',')' ',' '/'
DATA (NMOD(2,J),J=1,8)
- /'TIMB','ER (','GRAS','S AN','D UN','DERS','TORY',')' ',' '/'
DATA (NMOD(3,J),J=1,8)
- /'TALL','GRA','SS','2.5 ','FT (','75 C','M)' ',' '/'
DATA (NMOD(4,J),J=1,8)
- /'CHAP','ARRA','L, 6','FT ','(180','CM)' ',' '/'
DATA (NMOD(5,J),J=1,8)
- /'BRUS','H, 2','FT ','(60 ','CM)' ',' ',' ',' '/'
DATA (NMOD(6,J),J=1,8)
- /'DORM','ANT ','BRUS','H, H','ARDW','OOD ','SLAS','H' ',' '/'
DATA (NMOD(7,J),J=1,8)
- /'SOUT','HERN','ROU','GH ',' ',' ',' ',' ',' '/'
DATA (NMOD(8,J),J=1,8)
- /'CLOS','ED T','IMBE','R LI','TTER',' ',' ',' ',' '/'
DATA (NMOD(9,J),J=1,8)
- /'HARD','WOOD','LIT','TER ',' ',' ',' ',' ',' '/'
DATA (NMOD(10,J),J=1,8)
- /'TIMB','ER (','LITT','ER A','ND U','NDER','STOR','Y)' ',' '/'
DATA (NMOD(11,J),J=1,8)
- /'LIGH','T LO','GGIN','G SL','ASH ',' ',' ',' ',' '/'
DATA (NMOD(12,J),J=1,8)
- /'MEDI','UM L','OGGI','NG S','LASH',' ',' ',' ',' ',' '/'
DATA (NMOD(13,J),J=1,8)
- /'HEAV','Y LO','GGIN','G SL','ASH ',' ',' ',' ',' ',' '/'

```

C...

```

C... INPUT CHARACTER VARIABLES,
DATA INLINE /25* ' ', KEY/ ' ', ARG /' '
DATA NEEDED /10*.TRUE., 5*.FALSE., 15*.TRUE./

```

C...

```

C... INITIALIZE UNITS TO BASE FOR START UP
DATA IUNT /1/, METRIC/.FALSE./, ISLP/1/, PERCENT /.TRUE./

```

C... INITIALIZE STATIC POINTERS

```

DATA YEP,NOPE /.TRUE.,.FALSE./
DATA IBAS,IMET, LONG,ISHT,LIML,LIMH/1,2,1,2,1,2/
DATA IPCT, IDEG /1,2/
DATA TOBASE,NOT2BA /.TRUE.,.FALSE./
DATA IROS,IHUA,IFLE,IFLI,IRIN,ISCO,IMOR/1,2,3,4,5,6,7/
DATA IFUL,IWAF,ISLO,IREL,ISPC,IDBH,IHGT,ICRN
+ /8,9,10,11,12,13,14,15/
DATA I1HR,I10H,IHRB,IWDY,IW20,IWMD,IWDD,ISDD
+ /16,17,18,19,20,21,22,23/
DATA IW20B,I1DW,I1MD,IWGHT,IWDYL,IWLIV,ITMP /24,25,26,27,28,29,30/
DATA IUN1,IUN2,NON /31,32,33/

```

C... UNIT LABELS, RANGES, AND CONVERSION FACTOR FOR ROS (1)

```

DATA VARLBL(1) /'RATE OF SPREAD, '/
DATA ((UNTLBL(I,J,1),I=1,2),J=1,2)
1 /'CHAINS/HR','M/MIN','CH/H','M/MIN'/
DATA (RNG(I,1),I=1,2) /0.,250./
DATA CFAC(1)/0.33528/

```

C... UNIT LABELS, RANGES, AND CONVERSION FACTOR FOR HEAT/AREA (2)

```
DATA VARLBL(2) /'HEAT PER UNIT AREA, '/
DATA ((UNTLBL(I,J,2),I=1,2),J=1,2)
1 /'BTU/SQFT', 'KJ/SQM', 'BTU/SQFT', 'KJ/SQM'/
DATA (RNG(I,2),I=1,2) /0.,5000./
DATA CFAC(2) /11.35657/
```

C... UNIT LABELS, RANGES, AND CONVERSION FACTOR FOR FLAME LENGTH (3)

```
DATA VARLBL(3) /'FLAME LENGTH, '/
DATA ((UNTLBL(I,J,3),I=1,2),J=1,2)
1 /'FEET', 'METERS', 'FT', 'M'/
DATA (RNG(I,3),I=1,2) /0.,200./
DATA CFAC(3) /.3048/
```

C... UNIT LABELS, RANGES, AND CONVERSION FACTOR FOR FIRELINE INT (4)

```
DATA VARLBL(4) /'FIRELINE INTENSITY, '/
DATA ((UNTLBL(I,J,4),I=1,2),J=1,2)
1 /'BTU/FT/S', 'KWATTS/M', 'BTU/FT/S', 'KW/M'/
DATA (RNG(I,4),I=1,2) /0.,10000./
DATA CFAC(4) /3.46148/
```

C... UNIT LABELS, RANGES, AND CONVERSION FACTOR FOR REACTION INT (5)

```
DATA VARLBL(5) /'REACTION INTENSITY, '/
DATA ((UNTLBL(I,J,5),I=1,2),J=1,2)
1 /'BTU/SQFT/M', 'KWATT/SQM', 'BTU/SQFT/M', 'KW/SQM'/
DATA (RNG(I,5),I=1,2) /0.,10000./
DATA CFAC(5) /0.189276/
```

C... UNIT LABELS, RANGES, AND CONVERSION FACTOR FOR SCORCH HEIGHT (6)

```
DATA VARLBL(6) /'SCORCH HEIGHT, '/
DATA ((UNTLBL(I,J,6),I=1,2),J=1,2)
1 /'FEET', 'METERS', 'FT', 'M'/
DATA (RNG(I,6),I=1,2) /0.,500./
DATA CFAC(6) /.3048/
```

C... UNIT LABELS, RANGES, AND CONVERSION FACTOR FOR MORTALITY (7)

```
DATA VARLBL(7) /'TREE MORTALITY, '/
DATA ((UNTLBL(I,J,7),I=1,2),J=1,2)
1 /'PERCENT', 'PERCENT', '%', '%'/
DATA (RNG(I,7),I=1,2) /0.,100./
DATA CFAC(7) /1.0/
```

C... UNIT LABELS, RANGES, AND CONVERSION FACTOR FOR FUEL MODEL (8)

```
DATA VARLBL(8) /'FUEL MODEL'/
DATA ((UNTLBL(I,J,8),I=1,2),J=1,2)
1 /' ', ' ', ' ', ' '/
DATA (RNG(I,8),I=1,2) /1.,99./
DATA CFAC(8) /1.0/
```

C... UNIT LABELS, RANGES, AND CONVERSION FACTOR FOR WAF (9)

```
DATA VARLBL(9) /'EXPOSURE TO WIND, '/
DATA ((UNTLBL(I,J,9),I=1,2),J=1,2)
1 /'(CODE)', '(CODE)', '(CODE)', '(CODE)'/
DATA (RNG(I,9),I=1,2) /1.,5./
DATA CFAC(9) /1.0/
```

C... UNIT LABELS, RANGES, AND CONVERSION FACTOR FOR SLOPE (10)

```
DATA VARLBL(10) /'TERRAIN SLOPE, '/
DATA ((UNTLBL(I,J,10),I=1,2),J=1,2)
1 /'PERCENT', 'DEGREES', '%', 'DEG'/
DATA (RNG(I,10),I=1,2) /0.,100./
DATA CFAC(10) /-2.0/
```

C... UNIT LABELS, RANGES, AND CONVERSION FACTOR FOR 100 TO 10 HR (11)

```
DATA VARLBL(11) /'RELATION OF 100-HR TLFM TO 10-HR TLFM, '/
DATA ((UNTLBL(I,J,11),I=1,2),J=1,2)
1 /'PERCENT', 'PERCENT', '%', '%'/
DATA (RNG(I,11),I=1,2) /-10.,10./
DATA CFAC(11) /1.0/
```

C... UNIT LABELS, RANGES, AND CONVERSION FACTOR FOR TREE SPECIES (12)

```
DATA VARLBL(12) /'TREE SPECIES, '/
DATA ((UNTLBL(I,J,12),I=1,2),J=1,2)
1 /'(CODE)', '(CODE)', '(CODE)', '(CODE)'/
DATA (RNG(I,12),I=1,2) /1.,4./
DATA CFAC(12) /1.0/
```

C... UNIT LABELS, RANGES, AND CONVERSION FACTOR FOR DBH (13)

```
DATA VARLBL(13) /'TREE DBH, '/
DATA ((UNTLBL(I,J,13),I=1,2),J=1,2)
1 /'INCHES', 'CENTIMETER', 'IN', 'CM'/
DATA (RNG(I,13),I=1,2) /5.,40./
DATA CFAC(13) /2.54/
```

C... UNIT LABELS, RANGES, AND CONVERSION FACTOR FOR TREE HEIGHT (14)

```
DATA VARLBL(14) /'TREE HEIGHT, '/
DATA ((UNTLBL(I,J,14),I=1,2),J=1,2)
1 /'FEET', 'METERS', 'FT', 'M'/
DATA (RNG(I,14),I=1,2) /10.,300./
DATA CFAC(14) /.3048/
```

C... UNIT LABELS, RANGES, AND CONVERSION FACTOR FOR CROWN RATIO (15)

```
DATA VARLBL(15) /'CROWN RATIO, '/
DATA ((UNTLBL(I,J,15),I=1,2),J=1,2)
1 /'PERCENT', 'PERCENT', '%', '%'/
DATA (RNG(I,15),I=1,2) /10.,100./
DATA CFAC(15) /1.0/
```

C... UNIT LABELS, RANGES, AND CONVERSION FACTOR FOR 1 HR DEAD FM (16)

```
DATA VARLBL(16) /'1-HR FUEL MOISTURE, '/
DATA ((UNTLBL(I,J,16),I=1,2),J=1,2)
1 /'PERCENT', 'PERCENT', '%', '%'/
DATA (RNG(I,16),I=1,2) /1.,40./
DATA CFAC(16) /1.0/
```

C... UNIT LABELS, RANGES, AND CONVERSION FACTOR FOR 10 HR DEAD FM (17)

```
DATA VARLBL(17) /'10-HR FUEL MOISTURE, '/
DATA ((UNTLBL(I,J,17),I=1,2),J=1,2)
1 /'PERCENT', 'PERCENT', '%', '%'/
DATA (RNG(I,17),I=1,2) /1.,40./
DATA CFAC(17) /1.0/
```

C... UNIT LABELS, RANGES, AND CONVERSION FACTOR FOR LIVE HERB FM (18)
DATA VARLBL(18) /'LIVE HERBACEOUS MOISTURE, '/
DATA ((UNTLBL(I,J,18),I=1,2),J=1,2)
1 /'PERCENT', 'PERCENT', '%', '%'/
DATA (RNG(I,18),I=1,2) /30.,300./
DATA CFAC(18) /1.0/

C... UNIT LABELS, RANGES, AND CONVERSION FACTOR FOR LIVE WOODY FM (19)
DATA VARLBL(19) /'LIVE WOODY MOISTURE, '/
DATA ((UNTLBL(I,J,19),I=1,2),J=1,2)
1 /'PERCENT', 'PERCENT', '%', '%'/
DATA (RNG(I,19),I=1,2) /30.,300./
DATA CFAC(19) /1.0/

C... UNIT LABELS, RANGES, AND CONVERSION FACTOR FOR 20 FT WIND (20)
DATA VARLBL(20) /'20-FOOT WINDSPEED, '/
DATA ((UNTLBL(I,J,20),I=1,2),J=1,2)
1 /'MI/H', 'KM/H', 'MI/H', 'KM/H'/
DATA (RNG(I,20),I=1,2) /0.,30./
DATA CFAC(20) /1.609/

C... UNIT LABELS, RANGES, AND CONVERSION FACTOR FOR MIDLAME WIND (21)
DATA VARLBL(21) /'MIDFLAME WINDSPEED, '/
DATA ((UNTLBL(I,J,21),I=1,2),J=1,2)
1 /'MI/H', 'KM/H', 'MI/H', 'KM/H'/
DATA (RNG(I,21),I=1,2) /0.,30./
DATA CFAC(21) /1.609/

C... UNIT LABELS, RANGES, AND CONVERSION FACTOR FOR WIND DIRECT (22)
DATA VARLBL(22) /'WIND DIRECTION (FROM UPSLOPE), '/
DATA ((UNTLBL(I,J,22),I=1,2),J=1,2)
1 /'DEGREES', 'DEGREES', 'DEG', 'DEG'/
DATA (RNG(I,22),I=1,2) /0.,180./
DATA CFAC(22) /1.0/

C... UNIT LABELS, RANGES, AND CONVERSION FACTOR FOR SPREAD DIRECT (23)
DATA VARLBL(23) /'SPREAD DIRECTION (FROM HEAD FIRE), '/
DATA ((UNTLBL(I,J,23),I=1,2),J=1,2)
1 /'DEGREES', 'DEGREES', 'DEG', 'DEG'/
DATA (RNG(I,23),I=1,2) /0.,180./
DATA CFAC(23) /1.0/

C... UNIT LABELS, RANGES, AND CONVERSION FACTOR FOR 20 FT WIND ZOOM (24)
DATA VARLBL(24) /'20-FOOT WINDSPEED, '/
DATA ((UNTLBL(I,J,24),I=1,2),J=1,2)
1 /'MI/H', 'KM/H', 'MI/H', 'KM/H'/
DATA (RNG(I,24),I=1,2) /0.,99./
DATA CFAC(24) /1.609/

C... UNIT LABELS, RANGES, AND CONVERSION FACTOR FOR WEIGHTED DEAD
C... ZOOM (25)
DATA VARLBL(25) /'1-HR FUEL MOISTURE, '/
DATA ((UNTLBL(I,J,25),I=1,2),J=1,2)
1 /'PERCENT', 'PERCENT', '%', '%'/

```
DATA (RNG(I,25),I=1,2) /1.,60./
DATA CFAC(25) /1.0/
```

C... UNIT LABELS, RANGES, AND CONVERSION FACTOR FOR 1 HOUR ZOOM (26)

```
DATA VARLBL(26) /'1-HR FUEL MOISTURE, '/
DATA ((UNTLBL(I,J,26),I=1,2),J=1,2)
1 /'PERCENT','PERCENT','%','%'/
DATA (RNG(I,26),I=1,2) /1.,60./
DATA CFAC(26) /1.0/
```

C... UNIT LABELS, RANGES, AND CONVERSION FACTOR FOR WEIGHTED DEAD

C... ZOOM (27)

```
DATA VARLBL(27) /'1-HR FUEL MOISTURE, '/
DATA ((UNTLBL(I,J,27),I=1,2),J=1,2)
1 /'PERCENT','PERCENT','%','%'/
DATA (RNG(I,27),I=1,2) /1.,60./
DATA CFAC(27) /1.0/
```

C... UNIT LABELS, RANGES, AND CONVERSION FACTOR FOR LIVE HERB ZOOM (28)

```
DATA VARLBL(28) /'LIVE HERB MOISTURE, '/
DATA ((UNTLBL(I,J,28),I=1,2),J=1,2)
1 /'PERCENT','PERCENT','%','%'/
DATA (RNG(I,28),I=1,2) /30.,300./
DATA CFAC(28) /1.0/
```

C... UNIT LABELS, RANGES, AND CONVERSION FACTOR FOR WEIGHTED LIVE

C... ZOOM (29)

```
DATA VARLBL(29) /'LIVE WEIGHTED MOISTURE, '/
DATA ((UNTLBL(I,J,29),I=1,2),J=1,2)
1 /'PERCENT','PERCENT','%','%'/
DATA (RNG(I,29),I=1,2) /1.,60./
DATA CFAC(29) /1.0/
```

C... UNIT LABELS, RANGES, AND CONVERSION FACTOR FOR AIR TEMP (30)

```
DATA VARLBL(30) /'AIR TEMPERATURE, '/
DATA ((UNTLBL(I,J,30),I=1,2),J=1,2)
1 /'F','C','F','C'/
DATA (RNG(I,30),I=1,2) /32.,120./
DATA CFAC(30) /-1.0/
```

C... UNIT LABELS, RANGES, AND CONVERSION FACTOR FOR NON DIMENSIONS (33)

```
DATA ((UNTLBL(I,J,33),I=1,2),J=1,2)
1 /' ',' ',' ',' '/
DATA (RNG(I,33),I=1,2) /0.,99999./
DATA CFAC(33) /1.0/
```

C*****

C ASSIGN DEFAULT START UP VALUES

C

```
DATA WORDY,TWO,CHANGE,FIX,NTWO /.TRUE.,.FALSE.,.FALSE.,.FALSE.,1/
```

C

C FIRE BEHAVIOR/FIRE EFFECTS LIMITS

C*****

C.... PRESCRIPTION ROS

DATA (VK(1,J),J=1,3) /10.,20.,10./

DATA LIMITED(1) /.TRUE./

C.... PRESCRIPTION HPUA

DATA (VK(2,J),J=1,3) /10.,20.,10./

C.... PRESCRIPTION FL

DATA (VK(3,J),J=1,3) /2.,6.,4./

C.... PRESCRIPTION FIRELINE INTENSITY

DATA (VK(4,J),J=1,3) /10.,20.,10./

C.... PRESCRIPTION REACTION INTENSITY

DATA (VK(5,J),J=1,3) /10.,20.,10./

C.... PRESCRIPTION SCORCH HEIGHT

DATA (VK(6,J),J=1,3) /10.,20.,10./

C.... PRESCRIPTION MORTALITY

DATA (VK(7,J),J=1,3) /40.,60.,20./

C*****

C SITE DESCRIPTIONS

C*****

C.... FUEL MODEL — MEDIUM LOGGING SLASH

DATA MODL /12/

DATA LM1 /12/

DATA (VK(8,J),J=1,3) /12.,0.,-1./

DATA MOI /3*.TRUE.,2*.FALSE./

C... WIND ADJUSTMENT FACTOR FOR MODEL 12

DATA WAF /0.4/

DATA IEXPOS/1/

DATA (VK(9,J),J=1,3) /.4,0.,-1./

C.... SLOPE: 40 %

DATA (VK(10,J),J=1,3) /40.,0.,-1./

C.... 100 HR TO 10 HR RELATIONSHIP

DATA REL100 /0./

DATA (VK(11,J),J=1,3) /0.,0.,-1./

C.... TREE SPECIES

DATA (VK(12,J),J=1,3) /1.,0.,-1./

C.... DBH

DATA (VK(13,J),J=1,3) /8.,0.,-1./

C.... TREE HEIGHT

DATA (VK(14,J),J=1,3) /100.,0.,-1./

C.... CROWN RATIO

DATA (VK(15,J),J=1,3) /70.,0.,-1./

C*****

C... PRE SET ENVIRONMENTAL CONSTRAINTS

C*****

C... 1 HOUR CONSTRAINTS

DATA (VK(16,J),J=1,3) /1.,25.,24./

C... 10 HOUR CONSTRAINTS

DATA (VK(17,J),J=1,3) /1.,25.,24./

C... 100 HOUR CONSTRAINTS

C... DATA (VK(18,J),J=1,3) /1.,40.,39./

C... LIVE HERB CONSTRAINTS

DATA (VK(18,J),J=1,3) /30.,300.,270./

C... LIVE WOODY CONSTRAINTS

DATA (VK(19,J),J=1,3) /30.,300.,270./

C... 20 FOOT WIND SPEED CONSTRAINTS

DATA (VK(20,J),J=1,3) / 0.,30.,30./

C... MIDFLAME WIND SPEED CONSTRAINTS

DATA (VK(21,J),J=1,3) / 0.,25.,25./

C.... WIND DIRECTION

DATA (VK(22,J),J=1,3) /0.,180.,180./

C.... SPREAD DIRECTION

DATA (VK(23,J),J=1,3) /0.,180.,180./

C*****

C OUTPUT TABLE CONFIGURATION

C NO FIRE VARIABLE, DEAD TABLE VARIALBE = 10 HOUR, LIVE = HERB

C*****

DATA ITFB /0/

DATA TAB10 /.TRUE./

DATA TABHB /.TRUE./

C

END

SUBROUTINE CARX

C

C////PROGRAMMED BY LSB, SEM, 1988

C

C CALCULATIONS FOR KEYWORD = RUN

INCLUDE 'WINDCOM.INC'

INCLUDE 'IOCOM.INC'

INCLUDE 'UNITS.INC'

INCLUDE 'QSPREAD.INC'

INCLUDE 'DISPLAY.INC'

LOGICAL LOW,HIGH,YN

NONEIN = .TRUE.

MXD = -1

MXW = -1

LOD = 350

LOWI = 101

MAXWI = -1

MNLV = 350

MXLV = -1

NITER = 0

NUMLOW = 0

NUMHI = 0

ILIVE = 0

C

C INITIAL VALUES PRESCRIPTION TABLE VALUES BASED ON PARAMETERS

C

DO 5 J = 1,MX20

MAXMIN(J) = 0

MAXMAX(J) = 0

MAXWND(J) = 0

DO 5 I = 1,MXWD

DO 5 K = 1,MXS

INRX(J,I,K,1) = 0

INRX(J,I,K,2) = 999

INRX(J,I,K,3) = 0

INRX(J,I,K,4) = 999

INRX(J,I,K,5) = 0

INRX(J,I,K,6) = 999

INRX(J,I,K,7) = 0

INRX(J,I,K,8) = 25000

INRX(J,I,K,9) = 0

5 CONTINUE

C...

C... SLOPE IS CONSTANT

PSLOP = VK(ISLO,1)

C...

C... SET FUEL BED AND SLOPE CONSTANTS

CALL SETUP

CALL QSPREAD(1,2)

C...

C... GET LOWER WEIGHTED DEAD MOISTURE

```
IF (LIMITED(I1HR)) THEN
  FM(1) = VK(I1HR,1)
ELSE
  FM(1) = 1.
ENDIF
IF (LIMITED(I10H)) THEN
  FM(2) = VK(I10H,1)
ELSE
  FM(2) = 1.
ENDIF
FM(3) = FM(2) + REL100
CALL SETMOI
CALL QSPREAD(3,3)
LOWER = NINT(FCATMOIS(1)*100.)
```

C... GET UPPER WEIGHTED DEAD MOISTURE

```
IF (LIMITED(I1HR)) THEN
  FM(1) = VK(I1HR,2)
ELSE
  FM(1) = 30.
ENDIF
IF (LIMITED(I10H)) THEN
  FM(2) = VK(I10H,2)
ELSE
  FM(2) = 30.
ENDIF
FM(3) = FM(2) + REL100
CALL SETMOI
CALL QSPREAD(3,3)
IUPPER = NINT(FCATMOIS(1)*100.)
```

C... BEGIN MASTER DEAD FUEL MOISTURE LOOP

```
DO 100 IFM1 = LOWER, IUPPER
  JFM1 = IFM1
  XFM1 = FLOAT (IFM1)/100.
  FM(1) = XFM1/(FUELF(1,1)+(1.2427*(FUELF(2,1)+FUELF(3,1))))
  FM(1) = 100.*FM(1)
  IF (FM(1) .LT. 1.0) FM(1) = 1.0
  FM(2) = 1.28*FM(1)
  FM(3) = FM(2)+REL100
  IF (LIMITED(I1HR)) THEN
    IF (FM(1).LT.VK(I1HR,1) .OR. FM(1).GT.VK(I1HR,2)) GOTO 100
  ENDIF
  IF (MOI(2) .AND. LIMITED(I10H)) THEN
    IF (FM(2).LT.VK(I10H,1) .OR. FM(2).GT.VK(I10H,2)) GOTO 100
  ENDIF
  IF (MOI(3) .AND. LIMITED(I10H)) THEN
    IF (FM(3).LT.VK(I10H,1)+REL100 .OR.
    &      FM(3).GT.VK(I10H,2)+REL100) GOTO 100
  ENDIF
```

C... LIVE MOISTURE LOOP FOR TWO OR ONE LIVE FUEL COMPONENTS

```
IF (MOI(4) .OR. MOI(5)) THEN
  DO 40 ILIVE = 30,300,30
```

```

JLIVE = ILIVE
IF (MOI(4) .AND. MOI(5)) THEN
  XMF2 = FLOAT(ILIVE)/100.
  FM(4) = XMF2*100.
  FM(5) = FM(4)
  IF (LIMITED(IHRB)) THEN
    IF (FM(4) .LT. VK(IHRB,1)) GOTO 40
    IF (FM(4) .GE. VK(IHRB,2)) GOTO 100
  ENDIF
  IF (LIMITED(IWDY)) THEN
    IF (FM(5) .LT. VK(IWDY,1)) GOTO 40
    IF (FM(5) .GE. VK(IWDY,2)) GOTO 100
  ENDIF
ELSE
  XMF2 = FLOAT(ILIVE)/100.
  IF (MOI(4)) THEN
    FM(4) = FLOAT(ILIVE)
    IF (LIMITED(IHRB)) THEN
      IF (FM(4) .LT. VK(IHRB,1)) GOTO 40
      IF (FM(4) .GT. VK(IHRB,2)) GOTO 100
    ENDIF
  ELSE
    FM(5) = FLOAT(ILIVE)
    IF (LIMITED(IWDY)) THEN
      IF (FM(5) .LT. VK(IWDY,1)) GOTO 40
      IF (FM(5) .GT. VK(IWDY,2)) GOTO 100
    ENDIF
  ENDIF
ENDIF
ENDIF

```

C... SET MOISTURE PARAMETERS (LIVE AND DEAD FUELS) AND
C... BEGIN 20 FOOT WIND SPEED LOOP & REDUCE TO MIDFLAME

```

CALL SETMOI
CALL QSPREAD(3,3)
DO 30 IWIND = NINT(VK(IW20,1)), NINT(VK(IW20,2))
  JWIND = IWIND
  WMID = FLOAT(IWIND)*WAF
  IF (LIMITED(IWMD)) THEN
    IF (NINT(WMID) .LT. NINT(VK(IWMD,1))) GOTO 30
    IF (NINT(WMID) .GT. NINT(VK(IWMD,2))) GOTO 40
  ENDIF
  NITER = NITER + 1
  CALL TESTRX(LOW,HIGH,4,5,JFM1,JWIND+1,JLIVE)
  IF (LOW) THEN
    NUMLOW = NUMLOW+1
    GOTO 30
  ELSE IF (HIGH) THEN
    NUMHI = NUMHI+1
    GOTO 40
  ENDIF
CONTINUE
CONTINUE

```

CXXX

CXXX

30

40

ELSE

C... SET MOISTURE PARAMETERS (DEAD FUELS ONLY) AND

C... BEGIN 20 FOOT WIND SPEED LOOP & REDUCE TO MIDFLAME

C...

JLIVE=0

CALL SETMOI

CALL QSPREAD(3,3)

DO 50 IWIND = NINT(VK(IW20,1)), NINT(VK(IW20,2))

JWIND = IWIND

WMID = FLOAT(IWIND)*WAF

IF (LIMITED(IWMD)) THEN

IF (NINT(WMID) .LT. NINT(VK(IWMD,1))) GOTO 50

IF (NINT(WMID) .GT. NINT(VK(IWMD,2))) GOTO 100

ENDIF

NITER = NITER + 1

CALL TESTRX(LOW,HIGH,4,5,JFM1,JWIND+1,JLIVE)

IF (LOW) THEN

NUMLOW = NUMLOW+1

CXXX

GOTO 50

ELSE IF (HIGH) THEN

NUMHI = NUMHI+1

CXXX

GOTO 100

ENDIF

50 CONTINUE

ENDIF

100 CONTINUE

IF (NONEIN) THEN

PLOW = FLOAT(NUMLOW)/FLOAT(NITER)

PHIGH= FLOAT(NUMHI)/FLOAT(NITER)

WRITE(LU6,3090)

WRITE(LU6,3092) PLOW*100.,PHIGH*100.

IF (LOG) THEN

WRITE(LU4,3090)

WRITE(LU4,3092) PLOW*100.,PHIGH*100.

ENDIF

CALL WAITE

RETURN

ENDIF

3090 FORMAT(//' *****'

& /' * ALL COMBINATIONS ARE OUT OF PRESCRIPTION *'

& /' *****')

3092 FORMAT(/' PERCENT BELOW = ',F5.0,' PERCENT ABOVE = ',F5.0)

C...

C... SET STEP VALUES FOR INITIAL TABLE

C... AFTER LISTING INPUT FOR LOGGED RUNS

C...

C...

C... ENTRY POINT FOR "LAST WINDOW" KEY WORD TO RECONSTRUCT LAST RX

C...

ENTRY LAWIND

LAWOK = .TRUE.

CXXX IF (LOG) CALL LILOG

MAXCOL = 8

ISTEPW = 1

```
105 NUM = (MAXWI-LOWI)/ISTEPW + 1
```

```
IF (NUM. GT. MAXCOL) THEN
```

```
    ISTEPW = ISTEPW+1
```

```
    GOTO 105
```

```
ENDIF
```

```
ISTEPD = 1
```

```
IF (LIMITED(ISDD)) THEN
```

```
    ISTRT = VK(ISDD,1)
```

```
    ISTOP = VK(ISDD,2)
```

```
ELSE
```

```
    ISTRT = 0
```

```
    ISTOP = 180
```

```
ENDIF
```

```
C...
```

```
DO 150 I = ISTRT,ISTOP,90
```

```
    ISTEPD = 1
```

```
    ITYPE = I/90 + 1
```

```
    CALL SHOWRX(LOWI,MAXWI,ISTEPW,LOD,MXD,ISTEPD,ITYPE)
```

```
    IF (ISTEPD .NE. 1) GOTO 150
```

```
    CALL CONTRL('WIND',ITYPE)
```

```
    IF (QUIT) RETURN
```

```
150 CONTINUE
```

```
IF (MOI(2) .OR. (MOI(4).AND.MOI(5)) ) THEN
```

```
    WRITE(LU6,6080)
```

```
    CALL READYN(YN)
```

```
    IF (YN) CALL CAMOIS
```

```
ENDIF
```

```
6080 FORMAT(/' DO YOU WANT TO SEE THE MOISTURE TABLES FOR ',  
& 'THIS RUN ? (Y/N) ')
```

```
C...
```

```
RETURN
```

```
END
```

SUBROUTINE CAMOIS

```
C
C////PROGRAMMED BY LSB, SEM, 1988
C
C... WEIGHTED MOISTURE VALUE AND COMPONENTS FOR CURRENT RX TABLE
C
  INCLUDE 'WND.COM.INC'
  INCLUDE 'QSPREAD.INC'
  INCLUDE 'IOCOM.INC'
  INCLUDE 'UNITS.INC'
  INCLUDE 'DISPLAY.INC'

C...
  CALL ZERO

C...
  DO 100 I = NINT(VK(I1HR,1)),NINT(VK(I1HR,2))
    FM(1) = FLOAT(I)
    DO 100 J = NINT(VK(I10H,1)),NINT(VK(I10H,2))
      FM(2) = FLOAT(J)
      FM(3) = FM(2) + REL100
      CALL SETMOI
      CALL QSPREAD(3,3)
      IWMS = NINT(100*FCATMOIS(1))
      IWMS = MIN0(IWMS,MXF)
      IF (TAB10) THEN
        MSD(IWMS,I,1) = MIN0(J,MSD(IWMS,I,1))
        MSD(IWMS,I,2) = MAX0(J,MSD(IWMS,I,2))
      ELSE
        MSD(IWMS,J,1) = MIN0(I,MSD(IWMS,J,1))
        MSD(IWMS,J,2) = MAX0(I,MSD(IWMS,J,2))
      ENDIF
    100 CONTINUE

C...
C... COMPUTE START/STOP/STEP AND PRINT DEAD FUEL MOISTURE TABLE
C...
C...
C... ENTRY POINT FOR RECONSTRUCTING LAST DEAD FUEL MOISTURE TABLE
C...
  ENTRY LADEAD

  MAXCOL = 10
  ISTEP = 1
  IF (TAB10) THEN
    ISTRT = NINT(VK(I1HR,1))
    ISTOP = NINT(VK(I1HR,2))
  ELSE
    ISTRT = NINT(VK(I10H,1))
    ISTOP = NINT(VK(I10H,2))
  ENDIF
  105 NUM = (ISTOP-ISTRT)/ISTEP + 1
  IF (NUM.GT.MAXCOL) THEN
    ISTEP = ISTEP + 1
    GOTO 105
  ENDIF
```

```
CALL SHOWMD(LOD,MXD,1,ISTRT,ISTOP,ISTEP,NUM)
```

```
CALL CONTRL('DEAD',IDUM)
```

```
IF (QUIT) RETURN
```

```
C...  
C...REPEAT FOR LIVE FUEL MOISTURE IF NEEDED  
C...  
IF (.NOT. MOI(4) .AND. .NOT. MOI(5)) RETURN
```

```
CALL ZERO
```

```
C...  
DO 200 I = NINT(VK(IHRB,1)),NINT(VK(IHRB,2)),10  
  FM(4) = FLOAT(I)  
  DO 200 J = NINT(VK(IWDY,1)),NINT(VK(IWDY,2)),10  
    FM(5) = FLOAT(J)  
    CALL SETMOI  
    CALL QSPREAD(3,3)  
    IWMS = NINT(FCATMOIS(2)*10.)-2  
    IF (TABHB) THEN  
      IHB = I/10-2  
      MSD(IWMS,IHB,1) = MIN0(J,MSD(IWMS,IHB,1))  
      MSD(IWMS,IHB,2) = MAX0(J,MSD(IWMS,IHB,2))  
    ELSE  
      IHB = J/10 - 2  
      MSD(IWMS,IHB,1) = MIN0(I,MSD(IWMS,IHB,1))  
      MSD(IWMS,IHB,2) = MAX0(I,MSD(IWMS,IHB,2))  
    ENDIF  
  200 CONTINUE
```

```
C...  
C... COMPUTE START/STOP/STEP AND PRINT LIVE FUEL MOISTURE TABLE  
C...  
C...  
C... ENTRY POINT FOR RECONSTRUCTING LAST LIVE FUEL MOISTURE TABLE  
C...  
ENTRY LALIVE
```

```
  MAXCOL = 8  
  ISTEP = 10  
  IF (TABHB) THEN  
    ISTRT = NINT(VK(IWDY,1))  
    ISTOP = NINT(VK(IWDY,2))  
  ELSE  
    ISTRT = NINT(VK(IHRB,1))  
    ISTOP = NINT(VK(IHRB,2))  
  ENDIF  
205 NUM = (ISTOP-ISTRT)/ISTEP + 1  
  IF (NUM. GT. MAXCOL) THEN  
    ISTEP = ISTEP + 10  
    GOTO 205  
  ENDIF  
  
CALL SHOWML(MNLV,MXLV,10,ISTRT,ISTOP,ISTEP,NUM)
```

CALL CONTRL('LIVE',IDUM)

RETURN

END

```
SUBROUTINE CONTRL(FROM, ITYPE)
```

```
INCLUDE 'WND.COM.INC'
```

```
INCLUDE 'IO.COM.INC'
```

```
LOGICAL SETUNT
```

```
CHARACTER FROM*4, KW*6
```

```
QUIT = .FALSE.
```

```
10 WRITE (LU6,1010)
```

```
1010 FORMAT (/ ' DISPLAY KEYWORD? ')
```

```
IF (WORDY) WRITE (LU6,1020)
```

```
1020 FORMAT(' ENTER: ZOOM,REPLAY,CONTINUE, '
```

```
& /' WORDY, TERSE, PAUSE, NOPAUSE, COMMENT, ENGLISH, METRIC'
```

```
& /' PERCENT, DEGREES, LOG, NOLOG, STATUS, QUIT, LIST, HELP, KEY')
```

```
C
```

```
READ (LU5,1110) INLINE
```

```
1110 FORMAT (25A1)
```

```
CALL DCODE(INLINE, KEY, ARG)
```

```
C
```

```
C... INPUT (NOT ALLOWED)
```

```
IF (KEY(1:1).EQ. 'I') THEN
```

```
KW = 'INPUT'
```

```
WRITE(LU6,1115) KW, KW
```

```
GOTO 10
```

```
ENDIF
```

```
C... CHANGE (NOT ALLOWED)
```

```
IF (KEY(1:2).EQ. 'CH') THEN
```

```
KW = 'CHANGE'
```

```
WRITE(LU6,1115) KW, KW
```

```
GOTO 10
```

```
ENDIF
```

```
C... RUN (NOT ALLOWED)
```

```
IF (KEY(1:2).EQ. 'RU') THEN
```

```
KW = 'RUN'
```

```
WRITE(LU6,1115) KW, KW
```

```
GOTO 10
```

```
ENDIF
```

```
1115 FORMAT(1X, A6, ' NOT ALLOWED HERE. ENTER QUIT, THEN '
```

```
+ , A6, ' IF THAT IS WHAT YOU WANT.')
```

```
C... KEYWORD = LIST
```

```
IF (KEY(1:2).EQ. 'LI') THEN
```

```
CALL LIRX
```

```
GO TO 10
```

```
ENDIF
```

```
C
```

```
C... KEYWORD = ZOOM
```

```
IF (KEY(1:1).EQ. 'Z') THEN
```

```
IF (FROM .EQ. 'WIND') THEN
```

```
        CALL ZOOMRX(ITYPE)
    ELSE
        IF (FROM .EQ. 'DEAD') THEN
            CALL ZOOMMD
        ELSE
            CALL ZOOMML
        ENDIF
    ENDIF
    GO TO 10
ENDIF
```

C

```
C...  KEYWORD = REPLAY
      IF (KEY(1:2).EQ. 'RE') THEN
        IF (FROM .EQ. 'WIND') THEN
            CALL REPLRX(ITYPE)
        ELSE
            IF (FROM .EQ. 'DEAD') THEN
                CALL REPLMD
            ELSE
                CALL REPLML
            ENDIF
        ENDIF
        GO TO 10
    ENDIF
```

C

```
      IF (KEY(1:1) .EQ. 'H') THEN
        CALL HLPWIN(ARG, 'DISPLAY')
        GO TO 10
    ENDIF
```

C

```
      IF (KEY(1:1).EQ. 'K') THEN
        CALL KEY0
        CALL KEYALL
        GO TO 10
    ENDIF
```

C

```
      IF (KEY(1:1) .EQ. 'S') THEN
        CALL STATUS('DISP',WORDY,PAUSE,LOG,LOGOK,FILFLG)
        GOTO 10
    ENDIF
```

C

```
      IF (KEY(1:3) .EQ. 'COM') THEN
        CALL COMENT
        GO TO 10
    ENDIF
```

C

```
      IF (KEY(1:1).EQ. 'Q') THEN
        QUIT = .TRUE.
        RETURN
    ENDIF
```

C

```
      IF (KEY(1:3).EQ. 'CON') RETURN
```

C

```
      CALL KEYFLG
```

IF (OK) GO TO 10

C... CHECK FOR UNITS KEY WORDS

IF (SETUNT()) GOTO 10

C

WRITE (LU6,1130) KEY

1130 FORMAT (23H UNRECOGNIZED KEYWORD—,A4)

GO TO 10

END

```
SUBROUTINE FBTEST(IFB,VK,LOW,HIGH,V1,V2,V3,V4,V5,V6)
```

```
C...
```

```
C... SUBROUTINE TO TEST IF COMPUTED FIRE BEHAVIOR IS WITHIN
```

```
C... RX LIMITS
```

```
C...
```

```
C... 1 = ROS
```

```
C... 2 = HPUA
```

```
C... 3 = FLAME LENGTH
```

```
C... 4 = FIRELINE INT
```

```
C... 5 = REACTION INT
```

```
C... 6 = SCORCH HEIGHT (MORTALITY)
```

```
C...
```

```
LOGICAL LOW,HIGH
```

```
DIMENSION VK(33,3)
```

```
DIMENSION V(6)
```

```
V(1) = V1
```

```
V(2) = V2
```

```
V(3) = V3
```

```
V(4) = V4
```

```
V(5) = V5
```

```
V(6) = V6
```

```
C...
```

```
IF (NINT(V(IFB)) .LT. NINT(VK(IFB,1)) ) LOW = .TRUE.
```

```
IF (NINT(V(IFB)) .GT. NINT(VK(IFB,2)) ) HIGH = .TRUE.
```

```
RETURN
```

```
END
```

SUBROUTINE INRX

C

C////PROGRAMMED BY LARRY BRADSHAW, SEM, 1986

C

C

C.... INPUT AND CHANGE FOR KEYWORD = INPUT OR CHANGE

C

INCLUDE 'WND.COM.INC'

INCLUDE 'IO.COM.INC'

INCLUDE 'UNITS.INC'

INCLUDE 'QSPREAD.INC'

C

LOGICAL YN, WASMORT

CHARACTER CL*2,REL*30

DATA WASMORT /.FALSE./

IFB = 24

IDT = 25

ILT = 26

WASMORT = LIMITED(IMOR)

FIX = .FALSE.

10 IF (CHANGE) THEN

CALL READQ('CHANGE WHICH LINE ? (0-26)',

+ 0.,26.,NOPE,YEP,0,VAL,VK,QUIT)

IF (QUIT) GOTO 1000

LINE = IFIX(VAL)

IF (LINE .EQ. 0) GOTO 1000

ELSE

LINE = 0

ENDIF

C

C*****

C...THIS INPUT BLOCK IS FOR FIRE BEHAVIOR/EFFECTS -- LINES 1-7

C*****

6000 FORMAT(/' INPUT OF FIRE BEHAVIOR CONSTRAINTS... '/,

+ ' CONSTRAIN AT LEAST ONE OF LINES 1 - 7')

15 IF (LINE.EQ.0 .OR. LINE .LE. IMOR) THEN

IF (LINE .EQ. 0) THEN

WRITE(LU6,6000)

ISTRT = IROS

ISTOP = IMOR

ELSE

ISTRT = LINE

ISTOP = LINE

ENDIF

DO 500 I = ISTRT,ISTOP

WRITE(RNGSTR,'(I2)') I

READ(RNGSTR,'(A2)') CL

N = IEOS(VARLBL(I))

STR=CL//"—DO YOU WANT TO CONSTRAIN "//VARLBL(I)(1:N-1)//

+ " (Y/N) ?"

```

      L = IEOS(STR)
      CALL ASK2(L,STR,YEP)
      CALL READYN(YN)
      IF (YN) THEN
        LIMITED(I) = YEP
        CALL READIT('ENTER MINIMUM VALUE, ',
+           RNG(LIML,I),RNG(LIMH,I),NOPE,NOPE,-I,RLO,VK)
        CALL READIT('ENTER MAXIMUM VALUE, ',
+           RLO,RNG(LIMH,I),NOPE,NOPE,-I,RHI,VK)
        VK(I,1) = RLO
        VK(I,2) = RHI
        VK(I,3) = RHI-RLO
        IF (I.EQ. IMOR) THEN
          DO 450 J = ISPC,ICRN
450          NEEDED(J) = YEP
        ENDIF
      ELSE
        LIMITED(I) = NOPE
        IF (I.EQ. IMOR) THEN
          DO 460 J = ISPC,ICRN
460          NEEDED(J) = NOPE
        ENDIF
      ENDIF
500  CONTINUE
      ENDIF
      IF (FIX) GOTO 1000

C*****
C... THIS INPUT BLOCK IS FOR SITE DESCRIPTIONS
C*****

6010 FORMAT(/' INPUT OF SITE PARAMETERS... ')

25  CONTINUE
      IF (LINE.EQ.0 .OR. (LINE .GE. IFUL .AND. LINE .LE. ICRN)) THEN
        IF (LINE .EQ. 0) THEN
          WRITE(LU6,6010)
          ISTRT = IFUL
          ISTOP = ICRN
        ELSE
          ISTRT = LINE
          ISTOP = LINE
        ENDIF

      DO 680 I = ISTRT,ISTOP
        IF (I .EQ. IFUL) THEN
          CALL INFUEL
        ELSE
          IF (NEEDED(I) .OR. CHANGE) THEN
            IF (I .EQ. IWAF ) THEN
              WRITE(RNGSTR,'(I2)') I
              READ(RNGSTR,'(A2)') CL
              STR=CL/'—EXPOSURE OF FUELS TO THE WIND CODE ? 1-5'
620              CALL ASK2(IEOS(STR),STR,YEP)

```

```

      IF (WORDY) WRITE(LU6,7000)
      CALL WHAT(RNG(LIML,I),RNG(LIMH,I),
+         NOPE,YEP,0,VALUE,VK,OK,QUIT,NOPE)
      IF (.NOT. OK) GOTO 620
      WAF = SETWAF(VALUE)
      VK(I,1)=WAF
      VK(I,2)=0.
      VK(I,3)=-1.
      GOTO 680

```

```

ENDIF

```

```

IF (I .EQ. ISPC ) THEN

```

```

  WRITE(RNGSTR,'(I2)') I
  READ(RNGSTR,'(A2)') CL
  STR = CL// '—TREE SPECIES CODE ? 1-4'

```

```

640

```

```

  CALL ASK2(IEOS(STR),STR,YEP)
  IF (WORDY) WRITE(LU6,7050)

```

```

+

```

```

  CALL WHAT(RNG(LIML,I),RNG(LIMH,I),
      NOPE,YEP,0,VALUE,VK,OK,QUIT,NOPE)

```

```

  IF (.NOT. OK) GOTO 640

```

```

  VK(I,1)=VALUE

```

```

  VK(I,2)=0.

```

```

  VK(I,3)=-1.

```

```

  GOTO 680

```

```

ENDIF

```

```

WRITE(RNGSTR,'(I2)') I

```

```

READ(RNGSTR,'(A2)') CL

```

```

650

```

```

+

```

```

CALL READIT(CL// '—' //VARLBL(I),

```

```

      RNG(LIML,I),RNG(LIMH,I),NOPE,NOPE,-I,VALUE,VK)

```

```

  VK(I,1)=VALUE

```

```

  VK(I,2)=0.

```

```

  VK(I,3)=-1.

```

```

IF(I.EQ.IREL) THEN

```

```

  WRITE(LU6,660) VALUE

```

```

  CALL READYN(YN)

```

```

  IF (.NOT.YN) GOTO 650

```

```

  REL100 = VK(I,1)

```

```

ENDIF

```

```

ENDIF

```

```

ENDIF

```

```

680  CONTINUE

```

```

660  FORMAT(5X,'100-HR TLFM = 10-HR TLFM + ',F5.1,' %'. OK? (Y/N)')

```

```

      IF (VK(ISLO,1).GT.0 ) THEN

```

```

        NEEDED(IWDD) = .TRUE.

```

```

      ELSE

```

```

        NEEDED(IWDD) = .FALSE.

```

```

      ENDIF

```

```

ENDIF

```

```

7000  FORMAT ('

```

```

-      /'

```

```

-      /'

```

```

-      /'

```

```

-      /'

```

```

      1=EXPOSED'

```

```

      2=PARTIALLY SHELTERED'

```

```

      3=FULLY SHELTERED—OPEN STAND'

```

```

      4=FULLY SHELTERED—DENSE STAND'

```

```

      5=ENTER YOUR OWN WIND REDUCTION FACTOR')

```

```

7050 FORMAT ('          1=WESTERN LARCH OR DOUGLAS FIR'
-           /'          2=WESTERN HEMLOCK'
-           /'          3=ENGELMANN SPRUCE OR RED CEDAR'
-           /'          4=LODGEPOLE OR SUBALPINE FIR')

```

```

C*****
C...THIS INPUT BLOCK IS FOR PRE-SET ENVIRONMENTAL CONDITIONS (16-25)
C*****

```

```

6020 FORMAT(/' INPUT OF PRESET ENVIRONMENTAL LIMITS...'/
+ ' YOU MAY CONSTRAIN ANY OR NONE OF LINES 16-23')

```

```

30 CONTINUE

```

```

IF (LINE.EQ.0 .OR. (LINE .GE. I1HR .AND. LINE .LE. ISDD)) THEN
  IF (LINE .EQ. 0) THEN
    WRITE(LU6,6020)
    ISTRT = I1HR
    ISTOP = ISDD
  ELSE
    ISTRT = LINE
    ISTOP = LINE
  ENDIF

```

```

DO 700 I = ISTRT,ISTOP

```

```

  IF (NEEDED(I) .OR. CHANGE) THEN

```

```

    WRITE(RNGSTR,'(I2)') I

```

```

    READ(RNGSTR,'(A2)') CL

```

```

    N = IEOS(VARLBL(I))

```

```

    STR=CL//"—DO YOU WANT TO CONSTRAIN "//VARLBL(I)(1:N-1)//

```

```

    " (Y/N) ?"

```

```

    JPOS = INDEX(STR,'20-FOOT')

```

```

    IF (JPOS .NE. 0 .AND. METRIC) STR(JPOS:JPOS+6) = '6-METER'

```

```

    L = IEOS(STR)

```

```

    CALL ASK2(L,STR,YEP )

```

```

    CALL READYN(YN)

```

```

    IF (YN) THEN

```

```

      LIMITED(I) = YEP

```

```

      CALL READIT(

```

```

        'ENTER MINIMUM VALUE, ',

```

```

        RNG(LIML,I),RNG(LIMH,I),NOPE,NOPE,-I,RLO,VK)

```

```

      CALL READIT(

```

```

        'ENTER MAXIMUM VALUE, ',

```

```

        RLO,RNG(LIMH,I),NOPE,NOPE,-I,RHI,VK)

```

```

      VK(I,1) = RLO

```

```

      VK(I,2) = RHI

```

```

      VK(I,3) = RHI-RLO

```

```

    ELSE

```

```

      LIMITED(I) = NOPE

```

```

    ENDIF

```

```

  ENDIF

```

```

700 CONTINUE

```

```

ENDIF

```

```

C...
C... INPUT SECTION FOR OUTPUT TABLE CONFIGURATION

```

C...

```
40 CONTINUE
  IF (LINE.EQ.0 .OR.(LINE .GE. IFB .AND. LINE .LE. ILT)) THEN
    IF (LINE.EQ. 0) THEN
      WRITE(LU6,6025)
      ISTRT = IFB
      ISTOP = ILT
    ELSE
      ISTRT = LINE
      ISTOP = LINE
    ENDIF
    DO 790 I = ISTRT,ISTOP
      IF (I .EQ. IFB) THEN
        WRITE(LU6,6030)
        CALL READYN(YN)
        IF (YN) THEN
          STR = ' WHICH ONE ? 1-7'
750      CALL ASK2(IEOS(STR),STR,YEP)
          IF (WORDY) WRITE(LU6,6050)
          CALL WHAT(1.,7.,NOPE,YEP,0,VALUE,VK,OK,QUIT,NOPE)
          IF (.NOT. OK) GOTO 750
          ITFB = NINT(VALUE)
          IF (ITFB.EQ. IMOR) THEN
            DO 760 J = ISPC,ICRN
760      NEEDED(J) = YEP
          ENDIF
        ELSE
          ITFB = 0
        ENDIF
      ENDIF

      IF (I .EQ. IDT) THEN
        IF (.NOT. MOI(2)) GOTO 775
770      WRITE(LU6,6070)
6070  FORMAT(/' 25--DO YOU WANT 10-HR MOISTURE AS THE DEAD ',
1 'FUEL MOISTURE TABLE VALUE ? (Y-N)')
        CALL READYN(YN)
        IF (YN) THEN
          TAB10 = .TRUE.
          ITV = I10H
        ELSE
          TAB10 = .FALSE.
          ITV = I1HR
        ENDIF
        LS = IEOS(VARLBL(ITV))
        WRITE(LU6,6075) VARLBL(ITV)(1:LS-1)
6075  FORMAT(/' DEAD FUEL MOISTURE TABLE VALUE IS ',A,'. OK? (Y/N)')
        CALL READYN(YN)
        IF (.NOT. YN) GOTO 770
      ENDIF

775      IF (I .EQ. ILT) THEN
        IF (.NOT.MOI(4).AND..NOT.MOI(5).AND..NOT.CHANGE) GOTO 790
780      WRITE(LU6,6080)
6080  FORMAT(/' 26--DO YOU WANT LIVE HERBACEOUS MOISTURE AS THE LIVE ',
```

1 'FUEL MOISTURE TABLE VALUE ? (Y-N)')

CALL READYN(YN)

IF (YN) THEN

TABHB = .TRUE.

ITV = IHRB

ELSE

TABHB = .FALSE.

ITV = IWDY

ENDIF

LS = IEOS(VARLBL(ITV))

WRITE(LU6,6085) VARLBL(ITV)(1:LS-1)

6085 FORMAT(/' LIVE FUEL MOISTURE TABLE VALUE IS ',A,'. OK? (Y/N)')

CALL READYN(YN)

IF (.NOT. YN) GOTO 780

ENDIF

790 CONTINUE

ENDIF

6030 FORMAT(/' 24--INCLUDE A FIRE VARIABLE IN THE OUTPUT TABLE? (Y-N)')

6050 FORMAT(' 1=RATE OF SPREAD'/' 2=HEAT PER UNIT AREA'/'

1 ' 3=FLAME LENGTH'/' 4=FIRELINE INTENSITY'/'

2 ' 5=REACTION INTENSITY'/' 6=SCORCH HEIGHT'/'

3 ' 7=MORTALITY')

6025 FORMAT(/' SPECIFICATION OF OUTPUT TABLE DESIGN...')

C...

C... CHECK INPUT AND RETURN OR TRY AGAIN

IF (CHANGE) GOTO 10

C... CHECK FOR AT LEAST ONE FB VARIABLE

1000 DO 1010 I = IROS,IMOR

1010 IF (LIMITED(I)) GOTO 1050

WRITE(LU6,6005)

LINE = 0

FIX = .TRUE.

GOTO 15

6005 FORMAT(/' AT LEAST ONE OF LINES 1 - 7 MUST BE CONSTRAINED!'/)

C...

C... IF MORTALITY IS CONSTRAINED, GET SCORCH LIMITS

C...

1050 IF (LIMITED(IMOR)) THEN

SCLO = 0.

IF (VK(IMOR,1) .GT. 0) SCLO =

+ SCORLIM(NINT(VK(ISPC,1)),VK(IDBH,1),VK(ICRN,1)/100.,

+ VK(IHGT,1),VK(IMOR,1)/100.)

SCHI = SCORLIM(NINT(VK(ISPC,1)),VK(IDBH,1),VK(ICRN,1)/100.,

+ VK(IHGT,1),VK(IMOR,2)/100.)

ENDIF

7080 FORMAT(/' YOU HAVE CONSTRAINED BOTH ',A,' AND ',A,'.'

1 //' THERE IS A DIRECT RELATIONSHIP BETWEEN THEM.',1X,A,

2 //' 1. INPUT ',A18,' FROM ',F8.1,' TO ',F8.1,1X,A, /

- ' EQUALS ',A18,' FROM ',F8.1,' TO ',F8.1,1X,A, /

```

3  /' 2.  INPUT  ',A18,' FROM ',F8.1,' TO ',F8.1,1X,A, /
-  '      EQUALS ',A18,' FROM ',F8.1,' TO ',F8.1,1X,A)

```

```

7085 FORMAT(/ 1X,A,' NO LONGER DIRECTLY CONSTRAINED...')

```

```

7090 FORMAT(/'  INPUT  ',A18,' FROM ',F8.1,' TO ',F8.1,1X,A, /
-  '      EQUALS ',A18,' FROM ',F8.1,' TO ',F8.1,1X,A)

```

C...

C... CHECK FOR CONSTRAINTS OF BOTH FLAME LENGTH & FIRELINE INT.

```

IF (LIMITED(IFLE) .AND. LIMITED(IFLI) ) THEN

```

```

  L1 = IEOS(VARLBL(IFLE))-1

```

```

  L2 = IEOS(VARLBL(IFLI))-1

```

```

  REL = ' '

```

```

  L3 = IEOS(REL)

```

```

  V1 = CVRTU(VK(IFLE,1),CFAC(IFLE),METRIC,NOT2BA)

```

```

  V2 = CVRTU(VK(IFLE,2),CFAC(IFLE),METRIC,NOT2BA)

```

```

  V5 = CVRTU(VK(IFLI,1),CFAC(IFLI),METRIC,NOT2BA)

```

```

  V6 = CVRTU(VK(IFLI,2),CFAC(IFLI),METRIC,NOT2BA)

```

```

  V3 = (VK(IFLE,1)/.45)**2.17

```

```

  V3 = CVRTU(V3,CFAC(IFLI),METRIC,NOT2BA)

```

```

  V4 = (VK(IFLE,2)/.45)**2.17

```

```

  V4 = CVRTU(V4,CFAC(IFLI),METRIC,NOT2BA)

```

```

  V7 = 0.45*VK(IFLI,1)**.46

```

```

  V7 = CVRTU(V7,CFAC(IFLI),METRIC,NOT2BA)

```

```

  V8 = 0.45*VK(IFLI,2)**0.46

```

```

  V8 = CVRTU(V8,CFAC(IFLI),METRIC,NOT2BA)

```

```

WRITE(LU6,7080) VARLBL(IFLE)(1:L1),VARLBL(IFLI)(1:L2),REL(1:L3),

```

```

1  VARLBL(IFLE)(1:L1),V1,V2,UNTLBL(IUNT,1,IFLE),

```

```

2  VARLBL(IFLI)(1:L2),V3,V4,UNTLBL(IUNT,1,IFLI),

```

```

3  VARLBL(IFLI)(1:L2),V5,V6,UNTLBL(IUNT,1,IFLI),

```

```

4  VARLBL(IFLE)(1:L1),V7,V8,UNTLBL(IUNT,1,IFLE)

```

```

STR = 'WHICH DO YOU WANT ? (1-2) '

```

```

800 CALL ASK2(IEOS(STR),STR,YEP)

```

```

CALL WHAT(1.,2.,NOPE,YEP,0,VALUE,VK,OK,QUIT,NOPE)

```

```

IF (.NOT. OK) GO TO 800

```

```

IF (VALUE .EQ. 1.) THEN

```

```

  LIMITED(IFLI) = NOPE

```

```

  WRITE(LU6,7085) VARLBL(IFLI)(1:L2)

```

```

ELSE

```

```

  LIMITED(IFLE) = NOPE

```

```

  WRITE(LU6,7085) VARLBL(IFLE)(1:L1)

```

```

ENDIF

```

```

ENDIF

```

C... CHECK FOR CONSTRAINTS OF BOTH HPUA AND REACTION INTENSITY.

C...

```

IF (LIMITED(IHUA) .AND. LIMITED(IRIN) ) THEN

```

```

  L1 = IEOS(VARLBL(IHUA))-1

```

```

L2 = IEOS(VARLBL(IRIN))-1
REL = '(FOR A GIVEN FUEL MODEL). '
L3 = IEOS(REL)
CALL SETUP
CALL QSPREAD(1,1)
V1 = CVRTU(VK(IHUA,1),CFAC(IHUA),METRIC,NOT2BA)
V2 = CVRTU(VK(IHUA,2),CFAC(IHUA),METRIC,NOT2BA)
V5 = CVRTU(VK(IRIN,1),CFAC(IRIN),METRIC,NOT2BA)
V6 = CVRTU(VK(IRIN,2),CFAC(IRIN),METRIC,NOT2BA)

V3 = VK(IHUA,1)*SIGMA/384.
V3 = CVRTU(V3,CFAC(IHUA),METRIC,NOT2BA)
V4 = VK(IHUA,2)*SIGMA/384.
V4 = CVRTU(V4,CFAC(IHUA),METRIC,NOT2BA)

V7 = VK(IRIN,1)*384./SIGMA
V7 = CVRTU(V7,CFAC(IRIN),METRIC,NOT2BA)
V8 = VK(IRIN,2)*384./SIGMA
V8 = CVRTU(V8,CFAC(IRIN),METRIC,NOT2BA)

WRITE(LU6,7080) VARLBL(IHUA)(1:L1),VARLBL(IRIN)(1:L2),REL(1:L3),
1  VARLBL(IHUA)(1:L1),V1,V2,UNTLBL(IUNT,1,IHUA),
2  VARLBL(IRIN)(1:L2),V3,V4,UNTLBL(IUNT,1,IRIN),
3  VARLBL(IRIN)(1:L2),V5,V6,UNTLBL(IUNT,1,IRIN),
4  VARLBL(IHUA)(1:L1),V7,V8,UNTLBL(IUNT,1,IHUA)

```

```

810 STR = 'WHICH DO YOU WANT ? (1-2) '
CALL ASK2(IEOS(STR),STR,YEP)
CALL WHAT(1.,2.,NOPE,YEP,0,VALUE,VK,OK,QUIT,NOPE)
IF (.NOT. OK) GO TO 810

```

```

IF (VALUE .EQ. 1.) THEN
  LIMITED(IRIN) = NOPE
  WRITE(LU6,7085) VARLBL(IRIN)(1:L2)
ELSE
  LIMITED(IHUA) = NOPE
  WRITE(LU6,7085) VARLBL(IHUA)(1:L1)
ENDIF
ENDIF

```

C...

C... IF ONLY MORTALITY IS CONTRAINED, SET AND DISPLAY SCORCH VALUES

C...

```

IF (LIMITED(IMOR) .AND. .NOT.LIMITED(ISCO).AND..NOT.WASMORT ) THEN
  VK(SCO,1) = SCLO
  VK(SCO,2) = SCHI
  VK(SCO,3) = SCHI - SCLO
  L1 = IEOS(VARLBL(IMOR))-1
  L2 = IEOS(VARLBL(SCO))-1
  WRITE(LU6,7090)
1  VARLBL(IMOR)(1:L1),VK(IMOR,1),VK(IMOR,2),UNTLBL(IUNT,1,IMOR),
2  VARLBL(SCO)(1:L2),VK(SCO,1),VK(SCO,2),UNTLBL(IUNT,1,ISCO)
ENDIF

```

C... CHECK FOR CONSTRAINTS OF BOTH MORTALITY AND SCORCH HEIGHT.

C...

IF (LIMITED(ISCO) .AND. LIMITED(IMOR)) THEN

L1 = IEOS(VARLBL(ISCO))-1

L2 = IEOS(VARLBL(IMOR))-1

REL = '(GIVEN THE TREE DESCRIPTION.)'

L3 = IEOS(REL)

V1 = CVRTU(VK(ISCO,1),CFAC(ISCO),METRIC,NOT2BA)

V2 = CVRTU(VK(ISCO,2),CFAC(ISCO),METRIC,NOT2BA)

V5 = CVRTU(VK(IMOR,1),CFAC(IMOR),METRIC,NOT2BA)

V6 = CVRTU(VK(IMOR,2),CFAC(IMOR),METRIC,NOT2BA)

V3 = CVRTU(SCLO,CFAC(ISCO),METRIC,NOT2BA)

V4 = CVRTU(SCHI,CFAC(ISCO),METRIC,NOT2BA)

CALL MORT(NINT(VK(ISPC,1)),VK(IDBH,1),VK(ISCO,1),

1 VK(ICRN,1)/100.,VK(IHGT,1),PMORT,BARKT,CVS)

V7 = PMORT

CALL MORT(NINT(VK(ISPC,1)),VK(IDBH,1),VK(ISCO,2),

1 VK(ICRN,1)/100.,VK(IHGT,1),PMORT,BARKT,CVS)

V8 = PMORT

WRITE(LU6,7080) VARLBL(ISCO)(1:L1),VARLBL(IMOR)(1:L2),REL(1:L3),

1 VARLBL(ISCO)(1:L1),V1,V2,UNTLBL(IUNT,1,ISCO),

2 VARLBL(IMOR)(1:L2),V3,V4,UNTLBL(IUNT,1,IMOR),

3 VARLBL(IMOR)(1:L2),V5,V6,UNTLBL(IUNT,1,IMOR),

4 VARLBL(ISCO)(1:L1),V7,V8,UNTLBL(IUNT,1,ISCO)

STR = 'WHICH DO YOU WANT ? (1-2) '

820 CALL ASK2(IEOS(STR),STR,YEP)

CALL WHAT(1.,2.,NOPE,YEP,0,VALUE,VK,OK,QUIT,NOPE)

IF (.NOT. OK) GO TO 820

IF (VALUE .EQ. 1.) THEN

LIMITED(IMOR) = NOPE

WRITE(LU6,7085) VARLBL(IMOR)(1:L2)

ELSE

LIMITED(ISCO) = NOPE

WRITE(LU6,7085) VARLBL(ISCO)(1:L1)

VK(ISCO,1) = SCLO

VK(ISCO,2) = SCHI

VK(ISCO,3) = SCHI - SCLO

ENDIF

ENDIF

C... CHECK FOR CONSTRAINTS OF BOTH 20 FOOT AND MIDFLAME WIND

C...

IF (LIMITED(IW20) .AND. LIMITED(IWMD)) THEN

L1 = IEOS(VARLBL(IW20))-1

L2 = IEOS(VARLBL(IWMD))-1

REL = '(GIVEN FUEL EXPOSURE TO WIND.)'

L3 = IEOS(REL)

V1 = CVRTU(VK(IW20,1),CFAC(IW20),METRIC,NOT2BA)

V2 = CVRTU(VK(IW20,2),CFAC(IW20),METRIC,NOT2BA)

V5 = CVRTU(VK(IWMD,1),CFAC(IWMD),METRIC,NOT2BA)

V6 = CVRTU(VK(IWMD,2),CFAC(IWMD),METRIC,NOT2BA)

```
V3 = VK(IW20,1)*WAF
V3 = CVRTU(V3,CFAC(IW20),METRIC,NOT2BA)
V4 = VK(IW20,2)*WAF
V4 = CVRTU(V4,CFAC(IW20),METRIC,NOT2BA)
```

```
V7 = VK(IWMD,1)/WAF
V7 = CVRTU(V7,CFAC(IWMD),METRIC,NOT2BA)
V8 = VK(IWMD,2)/WAF
V8 = CVRTU(V8,CFAC(IWMD),METRIC,NOT2BA)
```

```
WRITE(LU6,7080) VARLBL(IW20)(1:L1),VARLBL(IWMD)(1:L2),REL(1:L3),
1  VARLBL(IW20)(1:L1),V1,V2,UNTLBL(IUNT,1,IW20),
2  VARLBL(IWMD)(1:L2),V3,V4,UNTLBL(IUNT,1,IWMD),
3  VARLBL(IWMD)(1:L2),V5,V6,UNTLBL(IUNT,1,IWMD),
4  VARLBL(IW20)(1:L1),V7,V8,UNTLBL(IUNT,1,IW20)
```

```
STR = 'WHICH DO YOU WANT ? (1-2) '
830  CALL ASK2(IEOS(STR),STR,YEP)
      CALL WHAT(1.,2.,NOPE,YEP,0,VALUE,VK,OK,QUIT,NOPE)
      IF (.NOT. OK) GO TO 830
```

```
IF (VALUE .EQ. 1.) THEN
  LIMITED(IW20) = NOPE
  WRITE(LU6,7085) VARLBL(IW20)(1:L2)
ELSE
  LIMITED(IWMD) = NOPE
  WRITE(LU6,7085) VARLBL(IWMD)(1:L1)
ENDIF
ENDIF
```

```
9999 RETURN
      END
```

SUBROUTINE LIRX

C

C////PROGRAMMED BY LARRY BRADSHAW, SEM, 1986

C

C

C LIST INPUT FOR KEYWORD = FIRE

C

C

INCLUDE 'WND.COM.INC'

INCLUDE 'IO.COM.INC'

INCLUDE 'UNITS.INC'

LOGICAL LOGGED

LOGGED = .FALSE.

LUOUT = LU6

1 CONTINUE

WRITE(LUOUT,6000)

6000 FORMAT(/' INPUT LIST FOR RXWINDOW')

C

WRITE(LUOUT,6020)

6020 FORMAT(/' FIRE BEHAVIOR CONSTRAINTS: '/')

DO 500 I = IROS,IMOR

IF (LIMITED(I)) THEN

CALL LISTIT(I,I,VK,LUOUT,VARLBL(I))

ELSE

CALL LISTIT(I,-1,VK,LUOUT,VARLBL(I))

ENDIF

500 CONTINUE

IF (.NOT. LOGGED) CALL WAITE

C

C*****

C SITE CONDITIONS

C*****

WRITE(LUOUT,6050)

6050 FORMAT(/' SITE CONDITIONS: '/')

DO 600 I = IFUL,ICRN

IF (I.EQ. IFUL) THEN

CALL LIFUEL(LUOUT)

NEEDED(IWAF) = .FALSE.

ELSE

IF (NEEDED(I)) CALL LISTIT(I,I,VK,LUOUT,VARLBL(I))

ENDIF

600 CONTINUE

NEEDED(IWAF) = .TRUE.

IF (.NOT. LOGGED) CALL WAITE

C*****

C PRESET ENVIRONMENTAL CONSTRAINTS

C*****

WRITE(LUOUT,6060)

```
6060 FORMAT(/' PRESET ENVIRONMENTAL CONSTRAINTS: '/')

DO 700 I = I1HR,ISDD
  IF (NEEDED(I)) THEN
    IF(LIMITED(I)) THEN
      CALL LISTIT(I,I,VK,LUOUT,VARLBL(I))
    ELSE
      CALL LISTIT(I,-1,VK,LUOUT,VARLBL(I))
    ENDIF
  ENDIF
700 CONTINUE

WRITE(LUOUT,6065)
6065 FORMAT(//' OUTPUT TABLE CONFIGURATIONS: '/')
IF (ITFB .NE. 0) THEN
  LS = IEOS(VARLBL(ITFB))
  WRITE(LUOUT,6070) VARLBL(ITFB)(1:LS),UNTLBL(IUNT,1,ITFB )
ELSE
  WRITE(LUOUT,6075)
ENDIF
6070 FORMAT(' 24—WINDOW FIRE BEHAVIOR TABLE VARIABLE: ',A,1X,A)
6075 FORMAT(' 24—WINDOW FIRE BEHAVIOR TABLE VARIABLE: NONE')

IF (MOI(2)) THEN
  ITV = I10H
  IF(.NOT. TAB10) ITV = I1HR
  LS = IEOS(VARLBL(ITV))
  WRITE(LUOUT,6080) VARLBL(ITV)(1:LS),UNTLBL(IUNT,2,ITV)
ELSE
  WRITE(LUOUT,6081) NINT(VK(IFUL,1))
ENDIF
6080 FORMAT(' 25—DEAD FUEL MOISTURE TABLE VARIABLE : ',A,1X,A)
6081 FORMAT(' 25—DEAD FUEL MOISTURE TABLE VARIABLE : NO TABLE FOR ',
1 'THIS FUEL MODEL (' ,I2,')')

IF (MOI(4) .AND. MOI(5)) THEN
  ITV = IHRB
  IF (.NOT. TABHB) ITV = IWDY
  LS = IEOS(VARLBL(ITV))
  WRITE(LUOUT,6085) VARLBL(ITV)(1:LS),UNTLBL(IUNT,2,ITV)
ELSE
  WRITE(LUOUT,6086) NINT(VK(IFUL,1))
ENDIF
6085 FORMAT(' 26—LIVE FUEL MOISTURE TABLE VARIABLE : ',A,1X,A)
6086 FORMAT(' 26—LIVE FUEL MOISTURE TABLE VARIABLE : NO TABLE FOR ',
1 'THIS FUEL MODEL (' ,I2,')')
IF (LOG .AND. .NOT. LOGGED) THEN
  LUOUT = LU4
  LOGGED = .TRUE.
  GOTO 1
ENDIF

RETURN
END
```

```
SUBROUTINE LISTIT(NV,L,VK2,LUOUT,IQU)
```

```
C$TEST
```

```
C
```

```
C/////PROGRAMMED BY PAT ANDREWS, NFFL, 1983
```

```
C
```

```
C
```

```
C .... LIST A LINE OF INPUT
```

```
C
```

```
INCLUDE 'WDCOM.INC'
```

```
INCLUDE 'IOCOM.INC'
```

```
C...
```

```
C *****
```

```
C * MODIFIED FOR SWITCHABLE UNITS BY *
```

```
C * LARRY BRADSHAW, SEM, 1987 *
```

```
C *****
```

```
C...
```

```
INCLUDE 'UNITS.INC'
```

```
C...
```

```
DIMENSION VK2(33,3),VAL(7)
```

```
CHARACTER*(*) IQU
```

```
CHARACTER*30 SKODE(4)
```

```
DATA SKODE/'W. LARCH/DOUG-FIR','W. HEMLOCK',
```

```
& 'ENGLEMAN SPRUCE/RED CEDAR','LODGEPOLE/SUBALPINE FIR'/
```

```
C
```

```
IF (VK2(NV,3).GT.0.) THEN
```

```
KL=IFIX(1.01+(VK2(NV,2)-VK2(NV,1))/VK2(NV,3))
```

```
DO 10 J=1,KL
```

```
VAL(J) = VK2(NV,1) + FLOAT(J-1)*VK2(NV,3)
```

```
VAL(J) = CVRTU(VAL(J),CFAC(NV),METRIC,NOT2BA)
```

```
10 CONTINUE
```

```
ELSE
```

```
VAL(1)=VK2(NV,1)
```

```
VAL(1) = CVRTU(VAL(1),CFAC(NV),METRIC,NOT2BA)
```

```
KL=1
```

```
ENDIF
```

```
C....
```

```
C.... TEST THE ITEM FOR A COMMA. IF THERE IS ONE, MODIFY IT
```

```
C... FOR UNITS. IF NO COMMA, ITEM IS NOT UNIT SENSITIVE
```

```
C...
```

```
NPOS = INDEX(IQU,',')
```

```
DO 20 I = 1,LEN(STR)
```

```
20 STR(I:I) = ' '
```

```
IF (NPOS .NE. 0) THEN
```

```
IUPT = IUNT
```

```
IF (NV .EQ. ISLO) IUPT = ISLP
```

```
LU = IEOS(UNTLBL(IUPT,ISHT,NV))
```

```
STR(1:NPOS+1) = IQU(1:NPOS+1)
```

```
STR(NPOS+2:) = UNTLBL(IUPT,ISHT,NV)(1:LU)
```

```
JPOS = INDEX(STR,'20-FOOT')
```

```
IF (JPOS .NE. 0 .AND. METRIC) STR(JPOS:JPOS+6) = '6-METER'
```

```
ELSE
```

```
STR(1:) = IQU(1:LEN(IQU))
```

```
ENDIF
```

```
CC...
```

```

IF (VAL(1).EQ. -1. .AND. NV .NE. ISCO .AND. NV .NE. IREL) THEN
  WRITE(LUOUT,3005) NV,STR
  RETURN
ENDIF

```

```

IF (L .LT. 0) THEN
  WRITE(LUOUT,3000) NV,STR
  RETURN
ENDIF

```

```

IF (L.GT. 0) THEN
  IF (CVRTU(VK2(NV,2),CFAC(NV),METRIC,NOT2BA) .LT. 1000.) THEN
    ASSIGN 2005 TO IFMT
    IF (KL .GT. 1) ASSIGN 2000 TO IFMT
  ELSE
    ASSIGN 2006 TO IFMT
    IF (KL .GT. 1) ASSIGN 2001 TO IFMT
  ENDIF

```

C...

```

IF (NV.LT.IWDD .AND. NV.NE.ISPC .AND. NV.NE.IREL) THEN
  WRITE (LUOUT,IFMT) L,STR,(VAL(J),J=1,KL)
ELSE
  IF(NV.EQ.ISPC) WRITE(LUOUT,4000)
+   L,STR,(VAL(J),J=1,KL),SKODE(NINT(VAL(1)))
  IF(NV.EQ.IREL) WRITE(LUOUT,4005) L,(VAL(J),J=1,KL)
  IF(NV.EQ.IWDD) WRITE(LUOUT,IFMT) L,STR,(VAL(J),J=1,KL)
  IF(NV.EQ.ISDD) WRITE(LUOUT,IFMT) L,STR,(VAL(J),J=1,KL)
ENDIF
ELSE
  ASSIGN 2003 TO IFMT
  IF ( CVRTU(VK2(NV,2),CFAC(NV),METRIC,NOT2BA) .LT. 1000.)
+   ASSIGN 2002 TO IFMT
  WRITE (LUOUT,IFMT) STR,(VAL(J),J=1,KL)
ENDIF

```

```
2000 FORMAT (1X,I2,'—',A40,1X,F6.1,' TO ',F6.1)
```

```
2001 FORMAT (1X,I2,'—',A40,1X,F6.1,' TO ' F6.0)
```

```
2002 FORMAT (5X,A40,1X,F6.1,' TO ',F6.1)
```

```
2003 FORMAT (5X,A24,1X,F6.1,' TO ' F6.0)
```

```
2005 FORMAT (1X,I2,'—',A40,1X,F6.1)
```

```
2006 FORMAT (1X,I2,'—',A40,1X,F6.0)
```

```
3000 FORMAT (1X,I2,'—',A40,' *** NOT CONSTRAINED ***')
```

```
3005 FORMAT (1X,I2,'—',A40,' *** UNINITIALIZED ***')
```

```
4000 FORMAT (1X,I2,'—',A40,1X,F6.0,2X,A25)
```

```
4005 FORMAT (1X,I2,'—100-HR TLFM = 10-HR TLFM + ',14X,F6.1,' %')
```

C

```

RETURN
END

```

```
SUBROUTINE MORT(ISPC,DBH,SCHT,RATIO,HEIGHT,PMORT,BARKT,CVS)
```

```
C...
```

```
C... SUBROUTINE TO COMPUTE PROBABILITY OF MORTALITY BASED ON MODEL  
C... BY RYAN AND REINHARDT.
```

```
C...
```

```
C... PROGRAMED BY LARRY S. BRADSHAW, SEM, 1988.
```

```
C... MODIFIED BY CAROLYN CHASE, IFSL, 1988
```

```
C... ALL VARIABLES ARE IMPLICITLY TYPED
```

```
C... PASSED VARIABLES:
```

```
C  ISPC  : SPIECES CODE(1=DF&WL;2=WH;3=ES&WRC,4=SF&LP)
```

```
C  DBH   : TREE DBH (INCHES)
```

```
C  SCHT  : SCORCH HEIGHT (FEET)
```

```
C  RATIO : CROWN RATIO
```

```
C  HEIGHT: TREE HEIGHT (FEET)
```

```
C
```

```
C... RETURNED VARIABLES
```

```
C  PMORT : COMPUTED MORTALITY (PERCENT)
```

```
C  BARKT : COMPUTED BARK THICKNESS
```

```
C...
```

```
C... INTERMEDIATES
```

```
C  DBHC  : DBH IN CENTIMETERS
```

```
C  BK    : BARK THICKNESS CONSTANT
```

```
C  CBH   : CROWN BASE HEIGHT
```

```
C  CLT   : LENGTH OF CROWN ONLY
```

```
C  CLS   : LENGTH OF CROWN SCORCHED
```

```
C  CVS   : VOLUME OF CROWN SCORCHED
```

```
C... GET DBH IN CENTIMETERS, COMPUTE BARK THICKNESS (CM), CONVERT TO
```

```
C... INCHES, AND COMPUTE BARK CONSTANT.
```

```
DBHC = 2.54*DBH
```

```
IF (ISPC .EQ. 1) THEN
```

```
  BARKT = 0.065*DBHC
```

```
ELSEIF (ISPC .EQ. 2) THEN
```

```
  BARKT = 0.056 + 0.043*DBHC
```

```
ELSEIF (ISPC .EQ. 3) THEN
```

```
  BARKT = 0.189 + 0.022*DBHC
```

```
ELSE
```

```
  BARKT = 0.015*DBHC
```

```
ENDIF
```

```
BARKT = BARKT/2.54
```

```
BK = 1.466 - 4.862*BARKT + 1.156*BARKT**2.
```

```
C... COMPUTE CROWN'S BASE HEIGHT, TOTAL LENGTH, AND SCORCHED LENGTH.
```

```
CBH = HEIGHT*(1.-RATIO)
```

```
CLT = HEIGHT - CBH
```

```
CLS = SCHT - CBH
```

```
C...
```

```
C... CONSTRAIN WITHIN PHYSICAL LIMITS
```

```
CLS = MAX1(CLS,0.)
```

```
IF (CLS .GT. CLT) CLS = HEIGHT
```

```
C...  
C... GET PERCENT OF CROWN VOLUME SCORCHED AND RESTRICT (0 TO 100)  
      CVS = (CLS*(2.*CLT-CLS)/CLT**2.)*100.  
CXXX      CVS = MAX1(CVS,0.)  
CXXX      CVS = MIN1(CVS,100.)  
  
C... COMPUTE MORTALITY (PERCENT)  
C...  
      PMORT = 1./(1.+EXP(-(BK+0.000535*CVS**2.)))  
      PMORT = MAX1(PMORT,0.)  
      PMORT = MIN1(PMORT,1.)  
      PMORT = PMORT*100.  
      RETURN  
      END
```

SUBROUTINE REPLMD

C

C////PROGRAMMED BY LSB, SEM, 1988

C

INCLUDE 'DISPLAY.INC'

C

CALL SHOWMD(LDSTRT,LDSTOP,LDSTEP,L1STRT,L1STOP,L1STEP,L1NUM)

RETURN

END

SUBROUTINE REPLML

C

C/////PROGRAMMED BY LSB, SEM, 1988

C

C

C CALCULATIONS FOR KEYWORD = RUN

INCLUDE 'DISPLAY.INC'

C

C

CALL SHOWML(LLSTRT,LLSTOP,LLSTEP,LWSTRT,LWSTOP,LWSTEP,LWNUM)

RETURN

END

```
SUBROUTINE REPLRX(ITYPE)
```

```
C
```

```
C/////PROGRAMMED BY LSB, SEM, 1988
```

```
C
```

```
C
```

```
C .... CALCULATIONS FOR KEYWORD = RUN
```

```
INCLUDE 'DISPLAY.INC'
```

```
C
```

```
C
```

```
CALL SHOWRX(LWMN,LWMX,LWST,LDMN,LDMX,LDST,ITYPE)
```

```
RETURN
```

```
END
```

```
FUNCTION SCORLIM(ISPC,DBH,RATIO,HEIGHT,PMORT)
```

```
C...
```

```
C... RETURNS SCORCH HIEGHT LIMIT BASED ON MODEL BY RYAN AND REINHARDT.
```

```
C...
```

```
C... PROGRAMED BY LARRY S. BRADSHAW, SEM, 1988.
```

```
C... ALL VARIABLES ARE IMPLICITLY TYPED
```

```
C... PASSED VARIABLES:
```

```
C  ISPC  : SPIECES CODE(1=DF,WL;2=WH;3=ES,WRC,4=SF,LP)
```

```
C  DBH   : TREE DBH (INCHES) (5 TO 25)
```

```
C  RATIO : CROWN RATIO (.1 TO .9)
```

```
C  HEIGHT: TREE HEIGHT (FEET) (0 TO 160)
```

```
C  PMORT : MORTALITY LIMIT (PERCENT)
```

```
C...
```

```
C... RETURNED VARIABLES
```

```
C  SCOTLIM: SCORCH HEIGHT (FEET) (0 TO 160)
```

```
C...
```

```
C... INTERMEDIATES
```

```
C  DBHC : DBH IN CENTIMETERS
```

```
C  BK    : BARK THICKNESS CONSTANT
```

```
C  CBH   : CROWN BASE HEIGHT
```

```
C  CVS   : VOLUME OF CROWN SCORCHED
```

```
C  CL    : CROWN LENGTH (HEIGHT-CBH)
```

```
C... GET DBH IN CENTIMETERS, COMPUTE BARK THICKNESS (CM), CONVERT TO
```

```
C... INCHES, AND COMPUTE BARK CONSTANT.
```

```
DBHC = 2.54*DBH
```

```
IF (ISPC .EQ. 1) THEN
```

```
  BARKT = 0.065*DBHC
```

```
ELSEIF (ISPC .EQ. 2) THEN
```

```
  BARKT = 0.056 + 0.043*DBHC
```

```
ELSEIF (ISPC .EQ. 3) THEN
```

```
  BARKT = 0.189 + 0.022*DBHC
```

```
ELSE
```

```
  BARKT = 0.015*DBHC
```

```
ENDIF
```

```
BARKT = BARKT/2.54
```

```
BK = 1.466 - 4.862*BARKT + 1.156*BARKT**2.
```

```
C... COMPUTE CROWN'S BASE HEIGHT, TOTAL LENGTH, AND SCORCHED LENGTH.
```

```
CBH = HEIGHT*(1.-RATIO)
```

```
CL = HEIGHT - CBH
```

```
C...
```

```
C... GET PERCENT OF CROWN VOLUME SCORCHED AND RESTRICT (0 TO 100)
```

```
IF (PMORT .EQ. 1.) THEN
```

```
  CVS = 100.
```

```
ELSEIF (PMORT.EQ. 0.) THEN
```

```
  CVS = 0.
```

```
ELSE
```

```
  CVS = (-LOG(1./PMORT-1.) - BK )
```

```
ENDIF
```

```
CVS = MAX1(CVS,0.)
```

$$CVS = (CVS/0.000535)**0.5$$
$$CVS = \text{MIN1}(CVS, 100.)$$

C... SCORCH HIEGHT

CXXX
$$\text{SCORLIM} = \text{ABS}(CL*(CVS/100.))**0.5 + CBH)$$
CXXX
$$\text{SCORLIM} = \text{ABS}(CL*(1.-CVS/100.))**0.5 + CBH)$$
CXXX
$$\text{SCORLIM} = \text{ABS}(\text{HEIGHT}*(1.-CVS/100.))**0.5 - CBH - \text{HEIGHT})$$
$$\text{SCORLIM} = \text{ABS}(CL*(1.-CVS/100.))**0.5 - CBH - CL)$$

RETURN

END

```
SUBROUTINE SHOWMD(IDSTRT,IDSTOP,IDSTEP,I1STRT,I1STOP,I1STEP,NUM)
```

```
C...
```

```
C... DISPLAYS MOISTURE TABLES
```

```
C...
```

```
INCLUDE 'WDCOM.INC'
```

```
INCLUDE 'QSPREAD.INC'
```

```
INCLUDE 'IOCOM.INC'
```

```
INCLUDE 'UNITS.INC'
```

```
INCLUDE 'DISPLAY.INC'
```

```
DIMENSION KCOL(10)
```

```
LOGICAL BLNK(10)
```

```
CXXX LOGICAL PRINT
```

```
CHARACTER*7 OUTVAL(10), BLNK7,DASH(10)
```

```
DATA BLNK7 /'      '/
```

```
DATA DASH /10*' —— '/
```

```
STR = 'TABLE RANGES ARE VALID 10-HR TLFM RANGES.'
```

```
IF (.NOT. TAB10) STR = 'TABLE VALUES ARE VALID 1-HR TLFM RANGES.'
```

```
LS = IEOS(STR)
```

```
WRITE(LU6,6000) STR(1:LS),100*FUELF(1,1)
```

```
IF (LOG) WRITE(LU4,6000) STR(1:LS),100.*FUELF(1,1)
```

```
IF (MOI(2)) THEN
```

```
WRITE(LU6,6005) 100.*FUELF(2,1)
```

```
IF (LOG) WRITE(LU4,6005) 100.*FUELF(2,1)
```

```
ENDIF
```

```
IF (MOI(3)) THEN
```

```
WRITE(LU6,6010) 100.*FUELF(3,1)
```

```
IF (LOG) WRITE(LU4,6010) 100.*FUELF(3,1)
```

```
ENDIF
```

```
6000 FORMAT(/, ' DEAD FUEL MOISTURE TABLE: ',A,
```

```
& //' MOISTURE WEIGHTING FACTORS ARE: 1 HOUR (' ,F5.1, ' %')'
```

```
6005 FORMAT(' 10 HOUR (' ,F5.1, ' %')'
```

```
6010 FORMAT(' 100 HOUR (' ,F5.1, ' %')'
```

```
6020 FORMAT(1X,I2,'-',I2)
```

```
6022 FORMAT(/A)
```

```
6030 FORMAT(2X,I2,1X,'% I',10A7 )
```

```
6035 FORMAT(' DEAD FM I',10(2X,I3,2X))
```

```
6040 FORMAT(' —— I',10A7)
```

```
DO 110 J = 1,NUM
```

```
110 KCOL(J) = I1STRT + (J-1)*I1STEP
```

```
STR= ' WGHTE I 1-HOUR FUEL MOISTURE, %'
```

```
IF(.NOT.TAB10) STR=' WGHTE I 10-HOUR FUEL MOISTURE, %'
```

```
WRITE(LU6,6022) STR
```

```
WRITE(LU6,6035) (KCOL(J),J=1,NUM)
```

```
WRITE(LU6,6040) (DASH(J),J=1,NUM)
```

```
NROWS = 9
```

```
IF (LOG) THEN
```

```
WRITE(LU4,6022) STR
```

```

WRITE(LU4,6035) (KCOL(J),J=1,NUM)
WRITE(LU4,6040) (DASH(J),J=1,NUM)
ENDIF

```

```

DO 140 I = IDSTRT,IDSTOP,IDSTEP

```

```

CXXX PRINT = .FALSE.

```

```

DO 130 J = 1,NUM

```

```

BLANK(J) = .TRUE.

```

```

I1H = I1STRT + (J-1)*I1STEP

```

```

ICOL(J,1) = 0

```

```

ICOL(J,2) = 0

```

```

IF (MSD(I,I1H,2) .NE. -999) THEN

```

```

CXXX PRINT = .TRUE.

```

```

BLANK(J) = .FALSE.

```

```

ICOL(J,1) = MSD(I,I1H,1)

```

```

ICOL(J,2) = MSD(I,I1H,2)

```

```

ENDIF

```

```

130 CONTINUE

```

```

C...

```

```

C... PRINT ROW IF THERE IS AT LEAST ONE NON-BLANK COLUMN

```

```

C...

```

```

CXXX IF (PRINT) THEN

```

```

DO 135 J = 1,NUM

```

```

IF (.NOT.BLANK(J)) THEN

```

```

WRITE(OUTVAL(J),6020) (ICOL(J,K),K=2,1,-1)

```

```

ELSE

```

```

OUTVAL(J) = BLNK7

```

```

ENDIF

```

```

135 CONTINUE

```

```

IF (PAUSE .AND. NROWS .GT. 20) THEN

```

```

CALL WAITE

```

```

WRITE(LU6,6022) STR

```

```

WRITE(LU6,6035) (KCOL(J),J=1,NUM)

```

```

WRITE(LU6,6040) (DASH(J),J=1,NUM)

```

```

NROWS = 4

```

```

ENDIF

```

```

WRITE(LU6,6030) I, (OUTVAL(J),J=1,NUM)

```

```

NROWS = NROWS + 1

```

```

IF (LOG) WRITE(LU4,6030) I, (OUTVAL(J),J=1,NUM)

```

```

CXXX ENDIF

```

```

140 CONTINUE

```

```

WRITE(LU6,6040) (DASH(J),J=1,NUM)

```

```

IF (LOG) WRITE(LU4,6040) (DASH(J),J=1,NUM)

```

```

C...

```

```

LDSTRT = IDSTRT

```

```

LDSTOP = IDSTOP

```

```

LDSTEP = IDSTEP

```

```

L1STRT = I1STRT

```

```

L1STOP = I1STOP

```

```

L1STEP = I1STEP

```

```

L1NUM = NUM

```

```

RETURN

```

END

```
SUBROUTINE SHOWML(LSTRT,LSTOP,LSTEP,IWSTRT,IWSTOP,IWSTEP,NUM)
```

```
C...
```

```
C... DISPLAYS MOISTURE TABLES
```

```
C...
```

```
INCLUDE 'WDCOM.INC'
```

```
INCLUDE 'QSPREAD.INC'
```

```
INCLUDE 'IOCOM.INC'
```

```
INCLUDE 'UNITS.INC'
```

```
INCLUDE 'DISPLAY.INC'
```

```
PARAMETER (MXC=8)
```

```
DIMENSION KCOL(8)
```

```
LOGICAL BLANK(MXC)
```

```
CXXX
```

```
LOGICAL PRINT
```

```
CHARACTER*8 OUTVA2(MXC), BLNK9,DASH9(MXC)
```

```
DATA BLNK9 / ' ' , '/'
```

```
DATA DASH9/MXC* '-----'/'
```

```
STR = 'TABLE VALUES ARE VALID LIVE HERBACEOUS RANGES.'
```

```
IF (.NOT. TABH8) STR='TABLE VALUES ARE VALID LIVE WOODY RANGES.'
```

```
LS = IEOS(STR)
```

```
WRITE(LU6,7000) STR(1:LS),100*FUELF(1,2)
```

```
WRITE(LU6,7005) 100.*FUELF(2,2)
```

```
IF (LOG) THEN
```

```
WRITE(LU4,7000) STR(1:LS),100.*FUELF(1,2)
```

```
WRITE(LU4,7005) 100.*FUELF(2,2)
```

```
ENDIF
```

```
7000 FORMAT(/' LIVE FUEL MOISTURE TABLE: ',A,
```

```
& /' MOISTURE WEIGHTING FACTORS ARE: WOODY (' ,F5.1,' %)')
```

```
7005 FORMAT(' HERBACEOUS (' ,F5.1,' %)')
```

```
7020 FORMAT(1X,I3,'-',I3)
```

```
7022 FORMAT(/A)
```

```
7030 FORMAT(1X,I3,1X,'% I',8A8)
```

```
7035 FORMAT(' LIVE FM I',8(3X,I3,2X))
```

```
7040 FORMAT(' ----- I',8A8)
```

```
STR = ' WGTED I LIVE WOODY MOISTURE, %'
```

```
IF (.NOT. TABH8)
```

```
+ STR=' WGTED I LIVE HERBACEOUS MOISTURE, %'
```

```
DO 210 J = 1,NUM
```

```
210 KCOL(J) = IWSTRT + (J-1)*IWSTEP
```

```
NROWS= 9
```

```
WRITE(LU6,7022) STR
```

```
WRITE(LU6,7035) (KCOL(J),J=1,NUM)
```

```
WRITE(LU6,7040) (DASH9(J),J=1,NUM)
```

```
IF (LOG) THEN
```

```
WRITE(LU4,7022) STR
```

```
WRITE(LU4,7035) (KCOL(J),J=1,NUM)
```

```
WRITE(LU4,7040) (DASH9(J),J=1,NUM)
```

ENDIF

DO 240 I = LSTRT,LSTOP,LSTEP

IWT = I/10-2

CXXX PRINT = .FALSE.

DO 230 J = 1,NUM

IHR = (IWSTRT + (J-1)*IWSTEP)/10 - 2

BLANK(J) = .TRUE.

ICOL(J,1) = 0

ICOL(J,2) = 0

IF (MSD(IWT,IHR,2) .NE. -999) THEN

CXXX PRINT = .TRUE.

BLANK(J) = .FALSE.

ICOL(J,1) = MSD(IWT,IHR,1)

ICOL(J,2) = MSD(IWT,IHR,2)

ENDIF

230 CONTINUE

C...

C... PRINT ROW IF NOT ALL BLANKS

C...

CXXX IF (PRINT) THEN

DO 235 J = 1,NUM

IF (.NOT. BLANK(J)) THEN

WRITE(OUTVA2(J),7020) (ICOL(J,K),K=2,1,-1)

ELSE

OUTVA2(J) = BLNK9

ENDIF

235 CONTINUE

IF (PAUSE .AND. NROWS .GT. 20) THEN

CALL WAITE

WRITE(LU6,7022) STR

WRITE(LU6,7035) (KCOL(J),J=1,NUM)

WRITE(LU6,7040) (DASH9(J),J=1,NUM)

NROWS= 4

ENDIF

CXXX ENDIF

C...

WRITE(LU6,7030) (IWT+2)*10, (OUTVA2(J),J=1,NUM)

NROWS = NROWS + 1

IF (LOG) WRITE(LU4,7030) (IWT+2)*10, (OUTVA2(J),J=1,NUM)

C...

C... NEXT ROW

C...

240 CONTINUE

WRITE(LU6,7040) (DASH9(J),J=1,NUM)

IF (LOG) WRITE(LU4,7040) (DASH9(J),J=1,NUM)

LLSTRT = LSTRT

LLSTOP = LSTOP

LLSTEP = LSTEP

LWSTRT = IWSTRT

LWSTOP = IWSTOP

LWSTEP = IWSTEP

LWNUM = NUM

C...

RETURN
END

```
SUBROUTINE SHOWRX(IBEGLW,IENDW,ISTEPW,IBEGD,IENDD,ISTEPD,ITYPE)
```

```
C
```

```
C/////PROGRAMMED BY LSB, SEM, 1988
```

```
C
```

```
C .... CALCULATIONS FOR KEYWORD = RUN
```

```
INCLUDE 'WND.COM.INC'
```

```
INCLUDE 'IO.COM.INC'
```

```
INCLUDE 'UNITS.INC'
```

```
INCLUDE 'DISPLAY.INC'
```

```
C
```

```
PARAMETER (MXC=8)
```

```
CHARACTER*8 DASHOU(MXC), DASH, BLANK8, OUTROW(6, MXC), EQOUT(MXC)
```

```
CHARACTER WNDLBL*55, FTYPE(3)*5, WKODE(5)*3
```

```
CHARACTER*6 RCODE(9)
```

```
CHARACTER EQ10*11, EQ8*8
```

```
DIMENSION KRP(4)
```

```
LOGICAL BLANK, SS, LIVE, KILO
```

```
DATA EQ10, EQ8 /'===== I', ' ====='/
```

```
DATA DASH, BLANK8/' -----', '      '/
```

```
DATA FTYPE /'HEAD', 'FLANK', 'BACK'/
```

```
DATA WKODE /' UP', ' QU', ' X ', ' QD', ' DN'/
```

```
DATA RCODE /'LV-FM', 'TEMP', 'ROS ', 'HPUA', 'FLAME', 'FL-INT', 'R-INT',
```

```
1 'SCORCH', 'T-MORT'/
```

```
SS = .FALSE.
```

```
KILO = .FALSE.
```

```
C...
```

```
C... GENERATE OUTPUT TABLE LOOP FOR WIND FROM LOW TO HI,
```

```
C... THEN OUTPUT A ROW. COLUMN HEADINGS ARE WGTED FUEL MOIST.
```

```
C...
```

```
C...
```

```
C... ENSURE THERE IS SOMETHING TO PRINT IN FLANK AND BACK TABLES
```

```
NC = IEOS(FTYPE(ITYPE))
```

```
DO 245 I = 1, 31
```

```
DO 245 J = 1, 30
```

```
245 IF (INRX(I, J, ITYPE, 1) .GT. 0) GOTO 248
```

```
WRITE(LU6, 3096) FTYPE(ITYPE)(1:NC)
```

```
3096 FORMAT(/1X, 'NO ', A, ' FIRE COMBINATIONS ARE IN PRESCRIPTION.')
```

```
ISTEPD = -1
```

```
CALL WAITE
```

```
RETURN
```

```
248 CONTINUE
```

```
C... WRITE TABLE HEADER
```

```
WNDLBL = '20 FOOT WIND SPEED/MIDFLAME WIND SPEED, MI/H'
```

```
IF (METRIC) WNDLBL =
```

```
& '6 METER WIND SPEED/MIDFLAME WIND SPEED, M/S'
```

```
WRITE(LU6, 3095) FTYPE(ITYPE)(1:NC), FTYPE(ITYPE)(1:NC), WNDLBL
```

```
3095 FORMAT(' WIND SPEEDS AND WEIGHTED FUEL MOISTURES THAT RESULT ',
```

```
1 'IN FIRE BEHAVIOR'/ ' WITHIN PRESCRIPTION CONSTRAINTS FOR A ',
```

```

2  '*** ',A,' FIRE ***', //,' *** ',A,' FIRE ***',T22,A)
  IF (LOG) WRITE(LU4,3095) FTYPE(ITYPE)(1:NC),FTYPE(ITYPE)(1:NC),
1    WNDLBL

```

```

  NCOL = 0
  DO 250 I =1,MXC
    DASHOU(I) = BLANK8
    EQOUT(I) = BLANK8
    OUTROW(5,I) = BLANK8
250 OUTROW(6,I) = BLANK8

  DO 300 I = IBEGW,IENDW,ISTEPW
    NCOL = NCOL + 1
    WRITE (OUTROW(5,NCOL),8010)
    & CVRTU(FLOAT(I-1),CFAC(IW20),METRIC,NOT2BA)
    WRITE (OUTROW(6,NCOL),8010)
    & CVRTU(FLOAT(I-1)*WAF,CFAC(IW20),METRIC,NOT2BA)
    DASHOU(NCOL) = DASH
    EQOUT(NCOL) = EQ8
300 CONTINUE
    WRITE(LU6,6040) EQ10,EQOUT,(OUTROW(5,I),I=1,MXC),
1 (OUTROW(6,I),I=1,MXC),EQ10,EQOUT
    IF (LOG)
1 WRITE(LU4,6040) EQ10,EQOUT,(OUTROW(5,I),I=1,MXC),
2 (OUTROW(6,I),I=1,MXC),EQ10,EQOUT
6040 FORMAT(1X,A11,8A8/' WEIGHTED I',8A8/' DEAD FM % I',8A8/1X,A11,
1 8A8)
C... LOOP FOR EFFECTIVE DEAD MOISTURE PREPARE OUTPUT ROW
C... COLUMN BY COLUMN (20 FOOT WIND)

```

```

  NROWS = 8
  DO 500 IROW = IBEGD,IENDD,ISTEPD
    NCOL = 0
    BLANK = .TRUE.
    DO 350 I = 1,MXC
      OUTROW(1,I) = BLANK8
      OUTROW(2,I) = BLANK8
      OUTROW(3,I) = BLANK8
350 OUTROW(4,I) = BLANK8
    DO 450 JCOL= IBEGW,IENDW,ISTEPW
      NCOL = NCOL + 1
      IF(INRX(JCOL,IROW,ITYPE,1) .NE. 0 ) THEN
        BLANK = .FALSE.
C... PREPARE OUTPUT ROW 1 WITH WIND DIRECTION RANGE
        NR = 1
        IWK1 = INRX(JCOL,IROW,ITYPE,4)/45 + 1
        IWK2 = INRX(JCOL,IROW,ITYPE,5)/45 + 1
        IF (IWK1 .EQ. 1 .AND. IWK2 .EQ. 5) THEN
          WRITE(OUTROW(NR,NCOL),8020)
        ELSE
          WRITE(OUTROW(NR,NCOL),8015) WKODE(IWK1),WKODE(IWK2)
        ENDIF

```

```

C...
C... IF THERE IS A LIVE FUEL COMPONENT, PREPARE OUTPUT ROW 2 WITH

```

C... ALLOWABLE LIVE FUEL MOISTURE RANGE

LIVE = .FALSE.

IF (MOI(4).OR.MOI(5)) LIVE = .TRUE.

IF (LIVE) THEN

ASSIGN 8000 TO IFMT

IF(INRX(JCOL,IROW,ITYPE,2) .EQ. MAXMIN(JCOL) .AND.

& INRX(JCOL,IROW,ITYPE,3) .EQ. MAXMAX(JCOL)) THEN

ASSIGN 8005 TO IFMT

SS = .TRUE.

ENDIF

NR = NR + 1

KRP(NR) = 1

WRITE(OUTROW(NR,NCOL),IFMT)

+ (INRX(JCOL,IROW,ITYPE,I),I=2,3)

ENDIF

C...

C... IF SCORCH HEIGHT IS IN THE RX, PREPARE TEMPERATURE RANGE ROW

IF (LIMITED(ISCO) .OR. LIMITED(IMOR)) THEN

NR = NR + 1

KRP(NR) = 2

ASSIGN 8000 TO IFMT

WRITE(OUTROW(NR,NCOL),IFMT)

+ (INRX(JCOL,IROW,ITYPE,I),I=6,7)

ENDIF

C...

C... IF A FIRE BEHAVIOR VALUE IS IN THE PRESCRIPTION, ADD A ROW

IF (ITFB .NE. 0) THEN

NR = NR + 1

KRP(NR) = ITFB +2

VAL1=CVRTU(FLOAT(INRX(JCOL,IROW,ITYPE,8)),

1 CFAC(ITFB),METRIC,NOT2BA)

VAL2=CVRTU(FLOAT(INRX(JCOL,IROW,ITYPE,9)),

1 CFAC(ITFB),METRIC,NOT2BA)

ASSIGN 8000 TO IFMT

IF (ITFB.EQ.IHUA.OR.ITFB.EQ.IFLI.OR.ITFB.EQ.IRIN) THEN

KILO = .TRUE.

VAL1 = VAL1/1000.

VAL2 = VAL2/1000.

IF (VAL1 .GE. 1.0 .AND. VAL2 .GE. 1.0) THEN

ASSIGN 8001 TO IFMT

ELSEIF (VAL1 .LT. 1.0 .AND. VAL2 .GE. 1.0) THEN

VAL1 = VAL1*10.

ASSIGN 8002 TO IFMT

ELSEIF (VAL1 .LT. 1.0 .AND. VAL2 .LT. 1.0) THEN

VAL1 = VAL1*10.

VAL2 = VAL2*10.

ASSIGN 8003 TO IFMT

ENDIF

ENDIF

WRITE(OUTROW(NR,NCOL),IFMT) NINT(VAL1),NINT(VAL2)

ENDIF

ENDIF

450 CONTINUE

C... DECODE FORMAT FOR LIVE RANGES

```

8000 FORMAT(1X,I3,'-',I3)
8001 FORMAT(1X,I2,'K-',I2,'K')
8002 FORMAT(1X,'.',I1,'K-',I2,'K')
8003 FORMAT(1X,'.',I1,'K-',I1,'K')
8005 FORMAT(1X,I3,'-',I3)
8010 FORMAT(2X,F4.1,2X)
8015 FORMAT(1X,A3,'-',A3)
8020 FORMAT(1X,' ANY ')

```

C...

C... OUTPUT NON BLANK ROWS AND CONTINUE UNTIL ALL ROWS DONE

```

IF (.NOT. BLANK) THEN

```

```

  NROWS = NROWS + NR + 1

```

```

  IF (PAUSE .AND. NROWS .GT. 20) THEN

```

```

    NROWS = 5 + NR

```

```

    CALL WAITE

```

```

    WRITE(LU6,6030) FTYPE(ITYPE)(1:NC),WNDLBL

```

```

    WRITE(LU6,6040) EQ10,EQOUT,(OUTROW(5,I),I=1,MXC),
1      (OUTROW(6,I),I=1,MXC),EQ10,EQOUT

```

```

  ENDIF

```

```

  WRITE(LU6,6052) IROW,(OUTROW(1,J),J=1,MXC)

```

```

  IF (LOG)WRITE(LU4,6052)IROW,(OUTROW(1,J),J=1,MXC)

```

```

  DO 480 KR = 2,NR

```

```

    IF(LOG) WRITE(LU4,6054)RCODE(KRP(KR)),
1      (OUTROW(KR,J),J=1,MXC)

```

```

480    WRITE(LU6,6054) RCODE(KRP(KR)),(OUTROW(KR,J),J=1,MXC)

```

```

    WRITE(LU6,6053) DASHOU

```

```

    IF (LOG)WRITE(LU4,6053) DASHOU

```

```

  ENDIF

```

```

6030 FORMAT(' *** ',A,' FIRE ***',T22,A)

```

```

6052 FORMAT(1X,I2,1X,'W-DIR I',8A8)

```

```

6053 FORMAT(1X,'----- I',8A8)

```

```

6054 FORMAT(4X,A6,' I',8A8)

```

```

500 CONTINUE

```

```

LWMN = IBEGW

```

```

LWMX = IENDW

```

```

LWST = ISTEPW

```

```

LDMN = IBEGD

```

```

LDMX = IENDD

```

```

LDST = ISTEPD

```

C... WRITE TABLE VALUE UNITS AS NEEDED..

```

  WRITE(LU6,7000)

```

```

  NC = IEOS(VARLBL(IWLTV))

```

```

  IF (LIVE) WRITE(LU6,7005)

```

```

1  VARLBL(IWLTV)(1:NC),UNTLBL(IUNT,1,IWLTV)

```

```

  NC = IEOS(VARLBL(ITMP))

```

```

  IF (LIMITED(ISCO) .OR. LIMITED(IMOR)) WRITE(LU6,7005)

```

```

1  VARLBL(ITMP)(1:NC),UNTLBL(IUNT,1,ITMP)

```

```

  IF (ITFB .NE. 0) THEN

```

```

      ASSIGN 7005 TO IFMT
      IF (KILO) ASSIGN 7010 TO IFMT
      NC = IEOS(VARLBL(ITFB ))
      WRITE(LU6,IFMT) VARLBL(ITFB )(1:NC),UNTLBL(IUNT,1,ITFB )
ENDIF

IF (LOG) THEN
  WRITE(LU4,7000)

  NC = IEOS(VARLBL(IWLV))
  IF (LIVE) WRITE(LU4,7005)
1    VARLBL(IWLV)(1:NC),UNTLBL(IUNT,1,IWLV)

  NC = IEOS(VARLBL(ITMP))
  IF (LIMITED(ISCO) .OR. LIMITED(IMOR)) WRITE(LU4,7005)
1    VARLBL(ITMP)(1:NC),UNTLBL(IUNT,1,ITMP)

  IF (ITFB .NE. 0) THEN
    ASSIGN 7005 TO IFMT
    IF (KILO) ASSIGN 7010 TO IFMT
    NC = IEOS(VARLBL(ITFB ))
    WRITE(LU4,IFMT) VARLBL(ITFB )(1:NC),UNTLBL(IUNT,1,ITFB )
  ENDIF
ENDIF

```

```

7000 FORMAT(1X,'UNITS/CODES FOR TABLE VALUES ARE: ' /
1 6X,'W=DIR (UP=UP SLOPE,QU=QUARTER-UP,X=CROSS,QD=QUARTER-DOWN',
2 ' ,DN=DOWN SLOPE)')
7005 FORMAT(6X,A,1X,A)
7010 FORMAT(6X,A,1X,A,' (.3K = 300, 3K = 3000, 30K = 30000)')
RETURN

```

C... WRITE FOOTNOTES

```

IF (LIVE) THEN
  WRITE(LU6,6060)
ELSE
  WRITE(LU6,6055)
ENDIF

IF (SS) WRITE(LU6,6065)
IF (PAUSE) CALL WAITE
IF (LOG) THEN
  IF (LIVE) THEN
    WRITE(LU4,6060)
  ELSE
    WRITE(LU4,6055)
  ENDIF
  IF (SS) WRITE(LU4,6065)
ENDIF

```

```

6055 FORMAT( ' ## = VALID EFFECTIVE WIND/MOISTURE CELLS IN RX.',
&          /'      BLANK CELLS ARE OUT OF PRESCRIPTION.')
6060 FORMAT( ' RANGES IN CELLS ARE VALID LIVE FUEL MOISTURES THAT IN,',

```

```
& /' COMBINATION WITH CORRESPONDING WIND/DEAD MOISTURE VALUES',  
& /' ARE IN PRESCRIPTION. BLANK CELLS ARE OUT OF PRESCRIPTION.')
```

6065 FORMAT(' * DUE TO LIMIT ON EFFECTIVE WIND ALL WINDS GREATER ',
& /' THAN "*" LINE ARE IN RX BUT FIRE BEHAVIOR DOES NOT CHANGE.')

```
RETURN  
END
```

```
SUBROUTINE TESTRX(LO,HIG,START,STOP,ID,IW,IL)
```

```
C...SUBROUTINE TO GET FIRE BEHAVIOR AND TEST AGAINST RX PARMS
```

```
INCLUDE 'WINDCOM.INC'  
INCLUDE 'UNITS.INC'  
INCLUDE 'QSPREAD.INC'  
INCLUDE 'DISPLAY.INC'
```

```
C...
```

```
LOGICAL LO,HIG  
INTEGER START,STOP
```

```
C...
```

```
C... TFB IS TABLE FIRE BEHAVIOR. TFB(1) = 0 AS ITFB = 0 FOR NONE)
```

```
C...
```

```
DIMENSION TFB(8)  
EQUIVALENCE (TFB(2),XXRATE),(TFB(3),XHPUA),(TFB(4),XFLAME),  
+ (TFB(5),XBYRAM),(TFB(6),XXIR),(TFB(7),XSCOR77),(TFB(8),PMORT)  
DO 5 I = 1,8  
5   TFB(I) = 0.
```

```
XHPUA = HPUA  
XXIR = XIR  
LO = .FALSE.  
HIG = .FALSE.
```

```
C... CALL TO SETMOI AND QSPREAD GET FIRE BEHAVIOR AT GIVEN FUEL
```

```
C... MOISTURE AND WIND SPEED LEVEL
```

```
CALL SETMOI  
CALL QSPREAD(START,STOP)
```

```
C...
```

```
C... GET FIRE BEHAVIOR FOR QUARTER, FLANK & BACK FIRES & TEST RX
```

```
C... IF SPREAD DIRECTION DOES NOT ELIMINATE THEM
```

```
C...
```

```
IWSTEP = 45  
IF (LIMITED(IWDD)) THEN  
    IWSTRT = VK(IWDD,1)  
    IWSTOP = VK(IWDD,2)  
ELSE  
    IWSTRT = 0  
    IWSTOP = 180  
ENDIF
```

```
DO 50 IWDIR = IWSTRT, IWSTOP, IWSTEP  
    WDIR = FLOAT(IWDIR)  
    CALL VECTOR(XDIR,XRATE,XBYRAM,XFLAME,XSCOR77)  
    IHEAD = NINT(XDIR)  
CXXX    IBACK = IHEAD + 180  
    IF (LIMITED(ISDD)) THEN  
        ISSTRT = NINT(VK(ISDD,1))+IHEAD  
        ISSTOP = NINT(VK(ISDD,2))+IHEAD  
    ELSE  
        ISSTRT = IHEAD  
        ISSTOP = IHEAD + 180  
    ENDIF  
    DO 45 ISDIR = ISSTRT, ISSTOP, 90
```

```
ISDS = (ISDIR - IHEAD)/90 + 1
SDIR = FLOAT(ISDIR)
CALL FLANK(XDIR,XRATE,XBYRAM,XFLAME,XSCOR77)
LO = .FALSE.
HIG = .FALSE.
```

C

```
DO 40 I = 1,6
  IF (LIMITED(I))CALL FBTEST(I,VK,LO,HIG,XRATE/1.1,HPUA,
&      XFLAME,XBYRAM,XXIR,XSCOR77)
  IF (LIMITED(IMOR)) CALL FBTEST(ISCO,VK,LO,HIG,XRATE/1.1,
&      HPUA,XFLAME,XBYRAM,XXIR,XSCOR77)
```

C...

```
C... IF FIRE BEHAVIOR IN THIS DIRECTION IS IN RX, FLAG IT.
```

C...

```
  IF (HIG .OR. LO) GOTO 45
```

```
40  CONTINUE
```

```
C... GET MORTALITY AT CURRENT FIRE BEHAVIOR AND RATE IN CH/HR
```

```
  IF (ITFB.EQ.IMOR) CALL MORT(NINT(VK(ISPC,1)),VK(IDBH,1),
```

```
1    XSCOR77,VK(ICRN,1)/100.,VK(IHGT,1),PMORT,BARKT,CVS)
```

```
  XXRATE=XRATE/1.1
```

```
  CALL WINNER(ID,IW,IL,ISDS,IWDIR,77,NINT(TFB(ITFB+1)) )
```

```
45  CONTINUE
```

```
50  CONTINUE
```

```
RETURN
```

```
END
```

SUBROUTINE TRANGE(SC77,SCLO,SCHI,TLO,THI)

C...

C... SUBROUTINE TO RETURN TEMPERATURE RANGES FOR A GIVEN RANGE OF

C... SCORCH

C...

C... SC77 : 77 DEGREE DAY COMPUTED SCORCH HEIGHT

C... SCLO : LOWER SCORCH LIMIT

C... SCHI : UPPER SCORCH LIMIT

C... TLO : LOWER TEMPERATURE LIMIT

C... THI : UPPER TEMPERATURE LIMIT

C...

C... RATIO : ADJUSTMENT FROM SC77 TO SC(T)

C...

C... GET TEMPERATURE FOR LOWER & UPPER SCORCH LIMITS

TLO = -99.

THI = -99.

IF (SC77 .LE. 0.) RETURN

TLO = 140.-63.*(SCHI/SC77)

IF (TLO .LE. 40) TLO = -99.

THI = 140.-63.*(SCLO/SC77)

IF (THI .GE. 80) THI = -99.

RETURN

END

```
SUBROUTINE WINNER(ID,IW,IL,ISDS,IWD,IT,IFBV)
```

```

C...
C... ID = WEIGHTED DEAD FUEL MOISTURE
C... IW = 20 FOOT WIND SPEED (PASSED AS VALUE +1 SO IW=1=NO WIND)
C... IL = WEIGHTED LIVE FUEL MOISTURE
C... ISD = FIRE SPREAD DIRECTION (0,45,90,135,180)
C... IWD = WIND DIRECTION (0,45,90,135,180)
C... ITL = LOW END TEMP FOR SCORCH
C... ITH = HIGH END TEMP FOR SCORCH

C...
      INCLUDE 'DISPLAY.INC'
C
C...
C... SUBROUTINE TO COUNT WINNERS
C...
C... INRX(I,J,K): COUNT AND RANGE ARRAY — INRX(51,30,7)
C... I = 20 FOOT WIND SPEED (0-50) (PASSED +1 SO J=1=NO WIND)
C... J = DEAD WEIGHTED FUEL MOISTURE (1-30)
C... K = 1: SLOT FOR COUNTER
C... K = 2: LOW WEIGHTED LIVE FUEL MOISTURE FOR WINNER AT I AND J
C... K = 3: HIGH WEIGHTED LIVE FUEL MOISTURE FOR WINNER AT I AND J
C... K = 4: LOW SPREAD DIRECTION FOR WINNER AT I AND J
C... K = 5: HIGH SPREAD DIRECTION FOR WINNER AT I AND J
C... K = 6: LOW SCORCH TEMP FOR WINNER AT I AND J
C... K = 7: HIGH SCORCH TEMP FOR WINNER AT I AND J
C...
C... JW : LOCAL COPY OF IW
C... MXD: MAXIMUM DEAD FUEL MOISTURE IN PRESCRIPTION
C... LOD: MINIMUM DEAD FUEL MOISTURE IN PESCRPTION
C... MXW: MAXIMUM 20 FOOT WIND IN PRESCRIPTION
C... LOWI: MINIMUM 20 FOOT WIND IN PRESCRIPTION
C... MXLV : MAXIMUM WEIGHTED LIVE FUEL MOISTURE IN OUTPUT TABLE
C... MNLV : MINIMUM WEIGHTED LIVE FUEL MOISTURE IN OUTPUT TABLE
C... MAXMIN(ID): HIGHEST MINIMUM LIVE FUEL MOIST FOR A GIVEN DEAD FM
C... MAXMAX(ID): HIGHEST MAXIMUM LIVE FUEL MOIST FOR A GIVEN DEAD FM
C...
C... MAKE SUBSCRIPTS FROM SPREAD DIRECTION VALUES
C...
CXXXX      ISDS = ISD/90 + 1

      JW = IW
C... UPDATE WINNER KOUNTNER
      INRX(JW,ID,ISDS,1) = INRX(JW,ID,ISDS,1) + 1
C...
C... DEAD FUEL MOISTURE RANGES
      MXD = MAX0(MXD,ID)
      LOD = MIN0(LOD,ID)
C...
C... WIND SPEED RANGES
      MAXWND(IW) = JW
CXXXX      IF (LWIND.EQ. 1) MAXWND(ID) = MIN0(MAXWND(ID),NINT(EWIND/88.))
      MAXWI = MAX0(MAXWND(JW),MAXWI)
      MXW = MAX0(MXW,JW)

```

LOWI = MIN0(LOWI,JW)

C... UPDATE LIVE FUEL MOISTURE RANGES IF NEEDED

IF (IL .NE. 0) THEN

INRX(JW,ID,ISDS,2) = MIN0(INRX(JW,ID,ISDS,2),IL)

INRX(JW,ID,ISDS,3) = MAX0(INRX(JW,ID,ISDS,3),IL)

MAXMIN(JW) = MAX0(MAXMIN(JW),INRX(JW,ID,ISDS,2))

MAXMAX(JW) = MAX0(MAXMAX(JW),INRX(JW,ID,ISDS,3))

MXLV = MAX0(INRX(JW,ID,ISDS,3),MXLV)

MNLV = MIN0(INRX(JW,ID,ISDS,2),MNLV)

ENDIF

C...

C... WIND DIRECTION RANGES (HEAD, QTR-UP, CROSS, QTR_DN, DOWN WRT SLOPE)

INRX(JW,ID,ISDS,4) = MIN0(INRX(JW,ID,ISDS,4),IWD)

INRX(JW,ID,ISDS,5) = MAX0(INRX(JW,ID,ISDS,5),IWD)

C... TEMPERATURE RANGES (FOR SCORCH HEIGHT)

INRX(JW,ID,ISDS,6) = MIN0(INRX(JW,ID,ISDS,6),IT)

INRX(JW,ID,ISDS,7) = MAX0(INRX(JW,ID,ISDS,7),IT)

C... FIRE BEHAVIOR VALUE (FOR OUTPUT TABLE)

IFBV = MIN0(IFBV,32000)

INRX(JW,ID,ISDS,8) = MIN0(INRX(JW,ID,ISDS,8),IFBV)

INRX(JW,ID,ISDS,9) = MAX0(INRX(JW,ID,ISDS,9),IFBV)

NONEIN = .FALSE.

RETURN

END

SUBROUTINE ZOOMMD

```
INCLUDE 'WVDCOM.INC'  
INCLUDE 'DISPLAY.INC'  
INCLUDE 'IOCOM.INC'  
INCLUDE 'UNITS.INC'
```

```
CHARACTER STR1*70
```

```
C... GET 1 OR 10 HOUR CORNERS
```

```
IF (TAB10) THEN
```

```
  STR1= 'ENTER 1-HR TLFM BOUNDARIES AND STEP SIZE, '
```

```
  IPNT = I1HR
```

```
ELSE
```

```
  STR1= 'ENTER 10-HR TLFM BOUNDARIES AND STEP SIZE, '
```

```
  IPNT = I10H
```

```
ENDIF
```

```
10 CALL READIT(STR1,
```

```
+   VK(IPNT,1),VK(IPNT,2),YEP,YEP,I1MD,10.,VK)
```

```
IF (VK(I1MD,3) .EQ. -1) THEN
```

```
  WRITE(LU6,6005)
```

```
  GOTO 10
```

```
ENDIF
```

```
6005 FORMAT(' *** INPUT REQUIRES START, STOP, AND STEP VALUES ***')
```

```
C... CHECK FOR MORE THAN TEN COLUMNS
```

```
IBEG1 = NINT(VK(I1MD,1))
```

```
IEND1 = NINT(VK(I1MD,2))
```

```
ISTEP1 = NINT(VK(I1MD,3))
```

```
NUM1 = (IEND1-IBEG1)/ISTEP1 + 1
```

```
IF (NUM1 .GT. 10) THEN
```

```
  WRITE(LU6,6000)
```

```
  GOTO 10
```

```
ENDIF
```

```
6000 FORMAT('/' MORE THAN TEN COLUMNS IN 1 HOUR INPUTS, TRY AGAIN...')
```

```
C... GET DEAD MOISTURE CORNERS
```

```
20 CALL READIT('ENTER WEIGHTED DEAD FM BOUNDARY AND STEP SIZE, ',
```

```
+   FLOAT(LOD),FLOAT(MXD),YEP,YEP,IWGHT,15.,VK)
```

```
IF (VK(IWGHT,3) .EQ. -1) THEN
```

```
  WRITE(LU6,6005)
```

```
  GOTO 20
```

```
ENDIF
```

```
IBEGD = NINT(VK(IWGHT,1))
```

```
IENDD = NINT(VK(IWGHT,2))
```

```
ISTEPD = NINT(VK(IWGHT,3))
```

```
CALL SHOWMD(IBEGD,IENDD,ISTEPD,IBEG1,IEND1,ISTEP1,NUM1)
```

RETURN
END

SUBROUTINE ZOOMML

```
INCLUDE 'WND.COM.INC'  
INCLUDE 'DISPLAY.INC'  
INCLUDE 'IO.COM.INC'  
INCLUDE 'UNITS.INC'
```

CHARACTER STR1*70

C... GET WOODY CORNERS

```
IF (TABHB) THEN  
  STR1= 'ENTER WOODY FUEL MOISTURE BOUNDARIES AND STEP SIZE, '  
  IPNT = IWDY  
ELSE  
  STR1= 'ENTER LIVE HERB FUEL MOISTURE BOUNDARIES AND STEP SIZE, '  
  IPNT = IHRB  
ENDIF
```

```
10 CALL READIT(STR1,  
+ VK(IPNT,1),VK(IPNT,2),YEP,YEP,IWDYL,8.,VK)
```

```
IF (VK(IWDYL,3) .EQ. -1) THEN  
  WRITE(LU6,6005)  
  GOTO 10  
ENDIF
```

6005 FORMAT(' *** INPUT REQUIRES START, STOP, AND STEP VALUES ***')

C... CHECK FOR MORE THAN TEN COLUMNS

```
IBEGW = NINT(VK(IWDYL,1))  
IENDW = NINT(VK(IWDYL,2))  
ISTEPW = NINT(VK(IWDYL,3))  
NUMW = (IENDW-IBEGW)/ISTEPW + 1
```

```
IF (NUMW .GT. 8) THEN  
  WRITE(LU6,6000)  
  GOTO 10  
ENDIF
```

6000 FORMAT('/' MORE THAN EIGHT COLUMNS IN WOODY INPUTS, TRY AGAIN...')

C... GET WEIGHTED LIVE MOISTURE CORNERS

```
20 CALL READIT('ENTER WEIGHTED LIVE MOISTURE BOUNDS AND STEP SIZE,  
+ ',FLOAT(MNLV),FLOAT(MXLV),YEP,YEP,IWL,15.,VK)  
IF (VK(IWL,3) .EQ. -1) THEN  
  WRITE(LU6,6005)  
  GOTO 20  
ENDIF
```

```
IBEGL = NINT(VK(IWL,1))  
IENDL = NINT(VK(IWL,2))  
ISTEPL = NINT(VK(IWL,3))
```

CALL SHOWML (IBEGL, IENDL, ISTEPL, IBEGW, IENDW, ISTEPW, NUMW)

RETURN

END

SUBROUTINE ZOOMRX(ITYPE)

INCLUDE 'WINDCOM.INC'
INCLUDE 'DISPLAY.INC'
INCLUDE 'IOCOM.INC'
INCLUDE 'UNITS.INC'

PARAMETER (MXC=8)

C... GET WIND CORNERS

10 CALL READIT('ENTER 20 FOOT WIND BOUNDS AND STEP SIZE, ',
+ FLOAT(LOWI-1),FLOAT(MAXWI-1),YEP,YEP,IW20B,FLOAT(MXC),VK)
IF (VK(IW20B,3) .EQ. -1) THEN
WRITE(LU6,6005)
GOTO 10
ENDIF
6005 FORMAT(' *** INPUT REQUIRES START, STOP, AND STEP VALUES ***')

IBEGW = NINT(VK(IW20B,1)) + 1
IENDW = NINT(VK(IW20B,2)) + 1
ISTEPW = NINT(VK(IW20B,3))

IF ((IENDW-IBEGW-1)/ISTEPW .GT. MXC) THEN
WRITE(LU6,6000)
GOTO 10
ENDIF

6000 FORMAT('/' MORE THAN EIGHT COLUMNS IN WIND INPUTS, TRY AGAIN...')

C... GET DEAD MOISTURE CORNERS

20 CALL READIT('ENTER WEIGHTED DEAD FM BOUNDS AND STEP SIZE, ',
+ FLOAT(LOD),FLOAT(MXD),YEP,YEP,I1DW,15.,VK)

IF (VK(I1DW,3) .EQ. -1) THEN
WRITE(LU6,6005)
GOTO 20
ENDIF

IBEGD = NINT(VK(I1DW,1))
IENDD = NINT(VK(I1DW,2))
ISTEPD = NINT(VK(I1DW,3))

CALL SHOWRX(IBEGW,IENDW,ISTEPW,IBEGD,IENDD,ISTEPD,ITYPE)

RETURN
END


```
SUBROUTINE ASK2(LEN,IQU,SKIP)
```

```
C$TEST
```

```
C
```

```
C////PROGRAMMED BY PAT ANDREWS, NFFL, 1983
```

```
C    REVISED BY CAROLYN CHASE, IFSL, 1985
```

```
C
```

```
C    .... ASK A ONE LINE QUESTION
```

```
C
```

```
    INCLUDE 'IOCOM.INC'
```

```
    CHARACTER*70 IQUX,IQU
```

```
    LOGICAL SKIP
```

```
C
```

```
    IF (LEN.GT.70) LEN=70
```

```
    IF (LEN.LT.1) LEN=1
```

```
    DO 10 I = 1,70
```

```
10  IQUX(I:I)= ' '
```

```
    IQUX(1:LEN)=IQU(1:LEN)
```

```
C    .... PRECEDE BY LINE SKIP IF NEEDED
```

```
    ASSIGN 6000 TO IFMT
```

```
    IF (SKIP) ASSIGN 6001 TO IFMT
```

```
    WRITE (LU6,IFMT) IQUX
```

```
6000 FORMAT (1X,A)
```

```
6001 FORMAT (/1X,A)
```

```
C
```

```
    RETURN
```

```
    END
```

```
SUBROUTINE CHECK(V,R1,R2,RNGOK,IGR,NVAR,VALUE,VK,OK)
```

```
C$TEST
```

```
C
```

```
C////PROGRAMMED BY PAT ANDREWS, NFFL, 1983
```

```
C
```

```
C
```

```
C .... CHECK INPUT.
```

```
C      STARTING VALUE, ENDING VALUE, STEP SIZE
```

```
C
```

```
      INCLUDE 'IOCOM.INC'
```

```
C...
```

```
C *****
```

```
C * MODIFIED FOR SWITCHABLE UNITS BY *
```

```
C * LARRY BRADSHAW, SEM, 1987 *
```

```
C *****
```

```
C...
```

```
      INCLUDE 'UNITS.INC'
```

```
C...
```

```
      DIMENSION V(3),VK(33,3),VAL(30)
```

```
      LOGICAL OK,RNGOK,IGR,YN
```

```
C
```

```
      OK=.TRUE.
```

```
C
```

```
      IF (VALUE .LE. 50. .AND. VALUE .GT. 0.) THEN
```

```
          MAXVAL = NINT(VALUE)
```

```
          MAXVAL = MIN0(15,MAXVAL)
```

```
          IF (MAXVAL .EQ. 0) MAXVAL = 7
```

```
      ENDIF
```

```
      IF (.NOT.RNGOK .AND. V(3).NE.-1.) THEN
```

```
          WRITE (LU6,1000)
```

```
1000  FORMAT (/ ' A RANGE OF VALUES IS NOT ALLOWED FOR THIS INPUT'
```

```
        - / ' VARIABLE. — TRY AGAIN.')
```

```
          OK = .FALSE.
```

```
          GO TO 999
```

```
      ENDIF
```

```
C
```

```
      IF (V(3).NE.-1. .AND. V(1).GE.V(2)) THEN
```

```
          WRITE (LU6,1010) V(1),V(2)
```

```
1010  FORMAT (/ ' STARTING VALUE = ',F10.1,
```

```
        - / ' ENDING VALUE   = ',F10.1,
```

```
        - / ' NOT ALLOWED — TRY AGAIN.')
```

```
          OK=.FALSE.
```

```
          GO TO 999
```

```
      ENDIF
```

```
C
```

```
      IF (V(3).EQ.0.) THEN
```

```
          WRITE (LU6,1012)
```

```
1012  FORMAT (/ ' STEP SIZE = 0. NOT ALLOWED. '
```

```
        - / ' TRY AGAIN.')
```

```
          OK=.FALSE.
```

```
          GO TO 999
```

```
      ENDIF
```

```
C
```

```
IF (IGR) THEN
  DO 12 I=1,3
  IF (I.GE.2 .AND. V(3).EQ.-1.) GO TO 13
  VV = FLOAT(IFIX(V(I)))
  IF (V(I).NE.VV) THEN
    WRITE (LU6,1013) V(I)
1013  FORMAT (/ ' NON-INTEGER VALUE = ',F10.1
    -      / ' DOES NOT MAKE SENSE. TRY AGAIN. ')
    OK=.FALSE.
    GO TO 999
  ENDIF
12  CONTINUE
13  CONTINUE
ENDIF

C
DO 15 I=1,2
IF (I.EQ.2 .AND. V(3).EQ.-1.) GO TO 16
RR1=R1-.01
RR2=R2+.01
IF (V(I).LT.RR1 .OR. V(I).GT.RR2) THEN
  WRITE(LU6,1015) V(I),R1,R2
1015  FORMAT (/ ' THE VALUE ',F8.1
  -      / ' IS OUTSIDE THE LEGAL RANGE ',F8.1,' TO ',F8.1
  -      / ' TRY AGAIN. ')
  OK = .FALSE.
  GO TO 999
ENDIF
15 CONTINUE
16 CONTINUE

C
IF (V(3).EQ.-1.) GO TO 45

C
KL = IFIX(1.00001+(V(2)-V(1))/V(3))
IF (KL.GT.MAXVAL) THEN
  KL = MAXVAL
  V(2) = V(1) + FLOAT(MAXVAL-1)*V(3)
  WRITE (LU6,1020) MAXVAL
1020  FORMAT (/ ' A MAXIMUM OF ',I2,' VALUES ARE ALLOWED. ')
ENDIF

C
DO 30 J=1,KL
VAL(J) = V(1) + FLOAT(J-1)*V(3)
30 CONTINUE

C
WRITE (LU6,1030) (VAL(J),J=1,KL)
1030  FORMAT (/ ' THE FOLLOWING VALUES WILL BE USED'/15F5.1)
WRITE (LU6,1040)
1040  FORMAT (' OK ? Y-N')
40  CALL READYN(YN)
IF (.NOT. YN) THEN
  OK = .FALSE.
  GOTO 999
ENDIF

C
45 CONTINUE
```

```
C
C... MAKE CONVERSION TO BASE UNITS FOR INDIVIDUAL VALUES
C... ZERO IS PASSED FOR SINGLE VALUES WITH NO UNIT DEPENDENCE
C... THE NEGATIVE OF THE VARIABLE NUMBER IS PASSED FOR SINGLE
C... VALUES WITH UNIT DEPENDENCE
C....
      IF (NVAR.EQ.0) THEN
        VALUE = CVRTU(V(1),CFAC(NON),METRIC,TOBASE)
      ELSE
        IF (NVAR .LT. 0) THEN
          VALUE = CVRTU(V(1),CFAC(IABS(NVAR)),METRIC,TOBASE)
        ELSE
          VK(NVAR,1) = CVRTU(V(1),CFAC(NVAR),METRIC,TOBASE)
          VK(NVAR,2) = CVRTU(V(2),CFAC(NVAR),METRIC,TOBASE)
          IF (IABS(NVAR) .EQ. 36) THEN
            VK(NVAR,3) = CVRTU(V(3),0.5555,METRIC,TOBASE)
          ELSE
            VK(NVAR,3) = CVRTU(V(3),CFAC(NVAR),METRIC,TOBASE)
          ENDIF
        ENDIF
      ENDIF
      ENDIF
C
999 CONTINUE
RETURN
END
```

SUBROUTINE CHKMOD

```
C
C////PROGRAMMED BY PAT ANDREWS, NFFL, 1983
C
C
C .... SEE IF FUEL MODEL (MODL) IS A STANDARD NFFL FUEL MODEL
C      OR A CUSTOM MODEL STORED IN A FILE -- LU7.
C
C .... IF A CUSTOM MODEL, CHANGE UNITS TO THOSE REQUIRED BY
C      SUBROUTINE SPREAD AND STORE PMOD AND NMOD, I=14 OR 15.
C
C
C      INCLUDE 'WND.COM.INC'
C
C      INCLUDE 'IO.COM.INC'
C
C      OK=.TRUE.
C
C      10 IF (MODL.LE.13) THEN
C          LM=MODL
C          GO TO 900
C      ENDIF
C
C      IF (.NOT.FILFLG) THEN
C          WRITE (LU6,5060)
C5060  FORMAT (/ ' A FUEL MODEL FILE IS NOT CURRENTLY ATTACHED TO THIS',
C          - ' RUN. ' )
C          GO TO 155
C      ENDIF
C
C      REWIND LU7
C      READ (LU7,7001,END=150) N1
C7001  FORMAT (A4)
C
C      80 READ (LU7,7000,END=150) MO,PMOD(15,13),(NMOD(15,J),J=1,8),
C          - (PMOD(15,J),J=1,5),(PMOD(15,J),J=9,11),(PMOD(15,J),J=6,8),
C          - PMOD(15,12)
C7000  FORMAT (I2,F1.1,8A4,5F4.2,F4.2,F5.0,F2.0,3F4.0,1X,F1.0)
C          IF (MO.EQ.MODL) GO TO 200
C          GO TO 80
C
C      150 WRITE (LU6,6000) MODL
C6000  FORMAT (/11H FUEL MODEL,I3,20H IS NOT IN THE FILE. )
C
C      155 CONTINUE
C
CYYY  155 WRITE (LU6,6005)
CYYY  6005 FORMAT (/47H DO YOU WANT TO TRY ANOTHER MODEL NUMBER ? Y-N      )
CYYY      CALL READYN(YES)
CYYY      IF (YES) THEN
CYYY          OK = .FALSE.
CYYY          GOTO 999
C
CXXX      CALL READIT('FUEL MODEL ? 1-99      ',
```

```
CXXX      -      1.,99.,.FALSE.,.TRUE.,0,X,DUM)
CXXX      MODL=X
CXXX      GO TO 10
CYYY      ENDIF
C
C... NEXT TWO STATEMENTS COMMENTED FOR USE IN WINDOWS, BY LSB
C      WRITE (LU6,6010)
C 6010 FORMAT (/ ' YOU SHOULD USE THE ''CUSTOM'' MODULE TO SPECIFY THE N'
C      -, 'AME OF' /' OR CHECK THE CONTENTS OF A FUEL MODEL FILE')
C      OK=.FALSE.
C      GO TO 999
C
C 200 CONTINUE
C .... CHECK VALIDITY OF CUSTOM FUEL MODEL PARAMETERS
C      DEPTH.GT.0, HEAT.GE.7000, MX.GE.10, 1H LOAD.GT.0
C      IF (PMOD(15,9).LT.0.01 .OR. PMOD(15,10).LT.7000. .OR.
C      -   PMOD(15,11).LT.10. .OR. PMOD(15,1).LT.0.001) THEN
C          WRITE (LU6,6030)
C 6030  FORMAT (/ ' THERE IS AN ERROR IN THE FUEL MODEL FILE.'
C      -/' YOU CAN USE ''CUSTOM'' TO LIST THE PARAMETERS FOR THE FUEL '
C      -, 'MODEL.'
C      -/' YOU CAN CHANGE A FUEL MODEL USING THE TSTMDL PROGRAM.')
C      OK=.FALSE.
C      GO TO 999
C      ENDIF
C
C .... LOAD FROM TONS/ACRE TO LBS/SQ.FT.
C      DO 220 J=1,5
C      PMOD(15,J) = PMOD(15,J)*.0459137
C 220 CONTINUE
C .... MAKE SURE SIGMA IS NOT ZERO
C      IF (PMOD(15,6).LT.99.) PMOD(15,6)=99.
C      IF (PMOD(15,7).LT.99.) PMOD(15,7)=99.
C      IF (PMOD(15,8).LT.99.) PMOD(15,8)=99.
C .... MOIS. OF EXT. FROM PERCENT TO FRACTION
C      PMOD(15,11) = PMOD(15,11)*.01
C
C      IF (NTWO.EQ.1) THEN
C          DO 240 J=1,15
C          PMOD(14,J) = PMOD(15,J)
C 240  CONTINUE
C          DO 250 J=1,8
C          NMOD(14,J) = NMOD(15,J)
C 250  CONTINUE
C          LM=14
C      ELSE
C          LM=15
C      ENDIF
C
C 900 IF (TWO .AND. NTWO.EQ.2) THEN
C      LM2=LM
C      ELSE
C          LM1=LM
C      ENDIF
C
```

CALL MFLAG

C

999 CONTINUE
RETURN
END

SUBROUTINE COMENT

```
C
C .... WRITTEN BY CAROLYN CHASE, IFSL, 1985
C
C .... WRITES USER'S COMMENT TO THE LOG FILE FOR PURPOSES OF
C      DOCUMENTATION
C
C      INCLUDE 'WND.COM.INC'
C
C      INCLUDE 'I.COM.INC'
C      CHARACTER*80 TELL
C
2000 FORMAT(/' NO LOG FILE CURRENTLY ACTIVE. ENTER LOG, THEN COMMENT.')
```

```
      IF (.NOT. LOG) THEN
        WRITE(LU6, 2000)
        RETURN
      ENDIF

      IF (WORDY) WRITE (LU6,5000)
5000 FORMAT (/1X,'ENTER TEXT FOR DOCUMENTATION.')
```

```
      -      /1X,' TO TERMINATE, ENTER ** ON A NEW LINE'
      -      /1X,' FOLLOWED BY A CARRIAGE RETURN.')
```

```
      WRITE (LU6,1000)
1000 FORMAT (//1X,75(1H*))
      -      /1X,'COMMENT:')
      IF (LOG) WRITE (LU4,1000)
C
      1 TELL(1:80)=' '
      READ (LU5,1020) TELL
1020 FORMAT (A80)
      IF (TELL(1:2).EQ.'**') GO TO 900
      IF (LOG) WRITE (LU4,1040) TELL
1040 FORMAT (1X,A80)
      GO TO 1
C
      900 CONTINUE
      WRITE (LU6,1060)
1060 FORMAT (1X,75(1H*))
      IF (LOG) WRITE (LU4,1060)
C
      RETURN
      END
```

SUBROUTINE CUSTOM

```
C$TEST
C
C////PROGRAMMED BY PAT ANDREWS, NFFL, 1983
C
C...
C *****
C *   MODIFIED FOR SWITCHABLE UNITS BY           *
C *   LARRY BRADSHAW, SEM, 1987                 *
C *****
C...
C   INCLUDE 'UNITS.INC'
C...
C .... ATTACH A FUEL MODEL FILE.
C   AND IF DESIRED, LIST THE FILE CONTENTS
C   OR PARAMETERS FOR A SPECIFIC FUEL MODEL
C
C   INCLUDE 'WINDCOM.INC'
C   INCLUDE 'IOCOM.INC'
C   CHARACTER MLBL(2,4)*8
C   LOGICAL ITSTHR,ITSOPN
C   DIMENSION N(18),NAME(8),PM(11),CMF(11),OM(11)
C
C   DATA ((MLBL(I,J),I=1,2),J=1,4)
C   + /'TON/AC','MTON/HA','1/FT','1/CM','FT','CM','BTU/LB','J/GM'/
C... CONVERSION FACTORS FOR FUEL PARAMETERS (1-5 - LOAD, 6 = DEPTH,
C... 7 = HEAT, 8 = EXT MOIST, 9-11 = SAV
C
C   DATA CMF /5*2.2417,30.48,2.3244,1.0,3*0.0328/
C
C .... THE CHARACTER 'A' IS WRITTEN WITH THE FUEL MODEL PARAMETERS TO
C   INDICATE THE FORMAT THAT IS USED FOR THE FUEL MODEL FILE.
C   THE FORMAT MAY CHANGE FOR UPDATES TO BEHAVE.
C   'A' INDICATES VERSION 2.0
C   DATA IFMT/1HA/
C
C .... IF A FILE IS NOT ATTACHED
C   IF (.NOT.FILFLG) THEN
C     WRITE (LU6,5090)
5090   FORMAT (/ ' A FUEL MODEL FILE IS NOT CURRENTLY ATTACHED TO THIS'
C   -   , ' RUN. ')
C     GO TO 50
C   ENDIF
C   WRITE (LU6,6000) FILNAM
6000   FORMAT (/ ' DO YOU WANT TO USE THE CURRENT FUEL MODEL FILE ? Y-N'
C   -   / '   FILE NAME: ',A12)
C   CALL READYN(YES)
C   IF (YES) GO TO 90
C   CLOSE (LU7)
50   WRITE (LU6,6010)
6010   FORMAT (/ ' FUEL MODEL FILE NAME ?')
C   READ (LU5,6020) FILNAM
6020   FORMAT (A12)
```

```
INQUIRE(FILE=FILNAM,EXIST=ITSTHR,OPENED=ITSOPN,ERR=57)
```

```
IF (ITSTHR) THEN
```

```
  GO TO 60
```

```
ELSE
```

```
  WRITE (LU6,6030)
```

```
6030  FORMAT (/ ' THERE IS NO FUEL MODEL FILE BY THIS NAME. ')
```

```
  FILFLG=.FALSE.
```

```
ENDIF
```

```
55 WRITE (LU6,6040)
```

```
6040  FORMAT (/ ' DO YOU WANT TO TRY ANOTHER FILE NAME ? Y-N ')
```

```
  CALL READYN(YES)
```

```
  IF (YES) THEN
```

```
    GO TO 50
```

```
  ELSE
```

```
    GO TO 999
```

```
ENDIF
```

```
57 WRITE (LU6,6045)
```

```
6045  FORMAT ( ' ILLEGAL FILE NAME.  TRY AGAIN. ')
```

```
  GO TO 50
```

```
C .... THE FILE EXISTS
```

```
60 IF (ITSOPN) GO TO 90
```

```
  OPEN(UNIT=LU7,FILE=FILNAM,STATUS='OLD',ACCESS='SEQUENTIAL',
```

```
  -   FORM='FORMATTED',ERR=70)
```

```
  FILFLG=.TRUE.
```

```
  GO TO 90
```

```
70 WRITE (LU6,6050)
```

```
6050  FORMAT (/ ' UNABLE TO OPEN REQUESTED FUEL MODEL FILE. ')
```

```
  FILFLG=.FALSE.
```

```
  GO TO 55
```

```
C
```

```
C .... CHECK THE CONTENTS OF THE FILE
```

```
90 CONTINUE
```

```
  REWIND LU7
```

```
  READ (LU7,7000,END=100) (N(I),I=1,18),IIFMT
```

```
7000  FORMAT (6X,18A4,1X,A1)
```

```
C
```

```
  WRITE (LU6,7020) FILNAM, (N(I),I=1,18)
```

```
7020  FORMAT (/ ' CUSTOM FUEL MODEL FILE NAME: ',A12
```

```
  -   / ' FILE DESCRIPTION: '/4X,18A4)
```

```
C
```

```
  IF (IIFMT.NE.IFMT) THEN
```

```
    WRITE (LU6,7010)
```

```
7010  FORMAT (/ ' THIS FILE IS IN AN OLD FORMAT AND CANNOT BE USED. '
```

```
  -   / ' USE THE PROGRAM 'REFORM' TO RESTRUCTURE YOUR FILE. '
```

```
  -   / ' (ON FCCC: @XQT BEHAVE*PROGRAMS.REFORM) ')
```

```
    GO TO 55
```

```
  ENDIF
```

```
C
```

```
  READ (LU7,7030,END=100) M,IIFMT
```

```
7030  FORMAT (I2,76X,A1)
```

```
  GO TO 150
```

```
C
```

```
100 WRITE (LU6,7040)
```

```
  IF (LOG) WRITE (LU4,7040)
```

```
7040  FORMAT (/ ' THE FILE IS EMPTY. ')
```

```
FILFLG=.FALSE.  
GO TO 55
```

```
C  
150 IF (IIFMT.NE.IFMT) THEN  
    WRITE (LU6,7010)  
    GO TO 55  
ENDIF  
READ (LU7,7030,END=200) M,IIFMT  
GO TO 150
```

```
C  
200 WRITE (LU6,7050)  
7050 FORMAT (/ ' DO YOU WANT A LIST OF THE FUEL MODELS THAT ARE STORED '  
-          / ' IN THE FILE ? Y-N' )  
CALL READYN(YES)  
IF (.NOT.YES) GO TO 300
```

```
C  
REWIND LU7  
READ (LU7,7000) (N(I),I=1,18)  
WRITE (LU6,7055) FILNAM, (N(I),I=1,18)  
IF (LOG) WRITE (LU4,7055) FILNAM, (N(I),I=1,18)  
7055 FORMAT (// ' CUSTOM FUEL MODEL FILE NAME: ',A12  
-          / ' FILE DESCRIPTION: '/4X,18A4  
-          //13X,'NUMBER FUEL MODEL NAME '/')  
NLINE=1
```

```
C  
250 READ (LU7,7060,END=300) M,(NAME(I),I=1,8),K  
7060 FORMAT (I2,1X,8A4,44X,I1)  
IF (K.EQ.1) THEN  
    WRITE (LU6,7070) M,(NAME(I),I=1,8)  
    IF (LOG) WRITE (LU4,7070) M,(NAME(I),I=1,8)  
7070 FORMAT (3X,'(DYNAMIC) ',I2,5X,8A4)  
ELSE  
    WRITE (LU6,7080) M,(NAME(I),I=1,8)  
    IF (LOG) WRITE (LU4,7080) M,(NAME(I),I=1,8)  
7080 FORMAT (3X,'(STATIC) ',I2,5X,8A4)  
ENDIF  
NLINE=NLINE+1  
IF (PAUSE .AND. NLINE.EQ.20) THEN  
    NLINE=1  
    CALL WAITE  
ENDIF  
GO TO 250
```

```
C  
300 WRITE (LU6,7090)  
7090 FORMAT (/ ' DO YOU WANT THE PARAMETER LIST FOR A SPECIFIC '  
-          / ' FUEL MODEL ? Y-N' )  
CALL READYN(YES)  
IF (.NOT.YES) GO TO 999
```

```
C  
CALL READIT('FUEL MODEL ? 14-99 '  
-          ,14.,99.,.FALSE.,.TRUE.,0,X,DUM)  
MX=X
```

```
C  
REWIND LU7  
READ (LU7,7000) (N(I),I=1,18)
```

```
C
  400 READ (LU7,8000,END=500) M,WF,(NAME(I),I=1,8),(PM(I),I=1,11),K
8000 FORMAT (I2,F1.1,8A4,5F4.2,F4.2,F5.0,F2.0,3F4.0,1X,I1)
      IF (M.NE.MX) GO TO 400

C
      IF (K.EQ.1) THEN
        WRITE (LU6,8010) M,(NAME(I),I=1,8)
        IF (LOG) WRITE (LU4,8010) M,(NAME(I),I=1,8)
8010  FORMAT (/' DYNAMIC CUSTOM MODEL ',I2,4H --- ,8A4)
      ELSE
        WRITE (LU6,8020) M,(NAME(I),I=1,8)
        IF (LOG) WRITE (LU4,8020) M,(NAME(I),I=1,8)
8020  FORMAT (/' STATIC CUSTOM MODEL ',I2,4H --- ,8A4)
      ENDIF

C
      WRITE (LU6,8030) FILNAM, (N(I),I=1,18)
      IF (LOG) WRITE (LU4,8030) FILNAM, (N(I),I=1,18)
8030  FORMAT (' FROM FILE NAME: ',A12
-         /' FILE DESCRIPTION: '/4X,18A4)

C...
C... MAKE UNIT CONVERSIONS
      DO 450 I = 1,11
450   OM(I) = CVRTU(PM(I),CMF(I),METRIC,NOT2BA)

C
      WRITE (LU6,8040) MLBL(IUNT,1),MLBL(IUNT,2),OM(1),OM(9),
+   MLBL(IUNT,3),OM(6),OM(2),OM(10),MLBL(IUNT,4),OM(7),
+   OM(3),OM(11),OM(8),OM(4),OM(5),WF
      IF (LOG) WRITE (LU4,8040) MLBL(IUNT,1),MLBL(IUNT,2),OM(1),OM(9),
+   MLBL(IUNT,3),OM(6),OM(2),OM(10),MLBL(IUNT,4),OM(7),
+   OM(3),OM(11),OM(8),OM(4),OM(5),WF
8040  FORMAT (/5X,'LOADS, ',A7,5X,'S/V RATIOS, ',A4,T55,'OTHER'
-         /3X,18(1H-),3X,17(1H-),3X,28(1H-)
-         /3X,'1 HR',9X,F5.2,3X,'1 HR',8X,F5.0,3X,'DEPTH, 'A2,
-         13X,F6.2
-         /3X,'10 HR',8X,F5.2,3X,'LIVE HERB',3X,F5.0,3X,
-         'HEAT CONTENT, 'A6,2X,F6.0
-         /3X,'100 HR',7X,F5.2,3X,'LIVE WOODY',2X,F5.0,3X,
-         'EXT MOISTURE (%)',9X,F3.0
-         /3X,'LIVE HERB',4X,F5.2,
-         /3X,'LIVE WOODY',3X,F5.2
-         //3X,'EXPOSED FUEL WIND ADJUSTMENT FACTOR =' ,F3.1)
      GO TO 300

C
      500 WRITE (LU6,9000) MX
9000  FORMAT (/' FUEL MODEL ',I2,' IS NOT IN THIS FILE.')
      GO TO 300

C
999  CONTINUE
      RETURN
      END
```

```
FUNCTION CVRTU(VALUE,CF,CONV,TOB)
```

```
C...
```

```
C... FUNCTION TO PERFORM UNIT CONVERSION
```

```
C... LARRY BRADSHAW, SEM 1987
```

```
C... VALUE = VALUE TO BE CONVERTED
```

```
C... CF      = CONVERSION FACTOR
```

```
C... CONV    = IF CONVERSION FLAG IS ON
```

```
C... TOB     = .TRUE.  PROCESS IS INPUT, COVERT FROM METRIC TO BASE
```

```
C... TOB     = .FALSE. PROCESS IS OUPUT, COVERT FROM BASE TO METRIC
```

```
C...
```

```
C... SPECIAL CONDITIONS: CF = -1: CONVERSION IS TEMPERATURE
```

```
C...                  CF = -2: CONVERSION IS SLOPE
```

```
C...                  VALUE = -1: UNINITIALIZED VALUE, NO CONVRT
```

```
INCLUDE 'UNITS.INC'
```

```
C...
```

```
LOGICAL CONV,TOB
```

```
CVRTU = VALUE
```

```
IF (VALUE .EQ. -1.) RETURN
```

```
IF (CONV) THEN
```

```
  IF (TOB) THEN
```

```
    IF (CF .GT. 0) CVRTU = VALUE/CF
```

```
    IF (CF .EQ. -1.) CVRTU = 1.8*VALUE + 32.
```

```
  ELSE
```

```
    IF (CF .GT. 0) CVRTU = VALUE*CF
```

```
    IF (CF .EQ. -1.) CVRTU = 0.5555*(VALUE-32.)
```

```
  ENDIF
```

```
ENDIF
```

```
C... CONVERT SLOPE
```

```
IF (.NOT. PERCENT) THEN
```

```
  IF (CF .EQ. -2) THEN
```

```
    IF (TOB) THEN
```

```
      CVRTU = ANINT(TAN(VALUE*0.01746)*100.)
```

```
    ELSE
```

```
      CVRTU = ANINT(ATAN(VALUE*0.01)*57.2727)
```

```
    ENDIF
```

```
  ENDIF
```

```
ENDIF
```

```
RETURN
```

```
END
```

```
SUBROUTINE DCODE(LINE,KEY,ARG)
```

```
C...DECODES LINE INTO KEYWORD AND ARGUEMENT
```

```
C...KEYWORD MUST BE AT LEAST FOUR CHARS IN LENGTH FOR ARG
```

```
C...TO BE RIGHT
```

```
C...
```

```
CHARACTER*1 LINE(25)
```

```
CHARACTER*4 KEY, ARG
```

```
CHARACTER*1 CHAR
```

```
KEY = ' '
```

```
ARG = KEY
```

```
DO 100 I = 1,25
```

```
  J = I
```

```
  IF (LINE(I) .EQ. ' ') GOTO 100
```

```
    KEY = LINE(I)//LINE(I+1)//LINE(I+2)//LINE(I+3)
```

```
    GOTO 150
```

```
100 CONTINUE
```

```
150 DO 200 I = J,25
```

```
  IF (LINE(I) .NE. ' ') GO TO 200
```

```
  K = I
```

```
  GOTO 250
```

```
200 CONTINUE
```

```
250 DO 300 I = K,25
```

```
  IF (LINE(I) .EQ. ' ') GOTO 300
```

```
    ARG = LINE(I)//LINE(I+1)//LINE(I+2)//LINE(I+3)
```

```
    GOTO 350
```

```
300 CONTINUE
```

```
C...
```

```
C... CONVERT TO UPPER CASE AND RETURN
```

```
C...
```

```
350 DO 400 I = 1,4
```

```
  IF (ICHAR(KEY(I:I)) .GE. 97 .AND. ICHAR(KEY(I:I)) .LE. 122) THEN
```

```
    IJK = ICHAR(KEY(I:I)) - 32
```

```
    KEY(I:I) = CHAR(IJK)
```

```
  ENDIF
```

```
  IF (ICHAR(ARG(I:I)) .GE. 97 .AND. ICHAR(ARG(I:I)) .LE. 122) THEN
```

```
    IJK = ICHAR(ARG(I:I))-32
```

```
    ARG(I:I) = CHAR(IJK)
```

```
  ENDIF
```

```
400 CONTINUE
```

```
RETURN
```

```
END
```

```
SUBROUTINE FLANK(XDIR,XRATE,XBYRAM,XFLAME,XSCOR77)
```

```
C
```

```
C////PROGRAMMED BY PAT ANDREWS, NFFL, 1983
```

```
C MODIFIED BY LARRY BRADSHAW, SEM, 1988 FOR RXWINDOW APPLICATION
```

```
C
```

```
C .... FIND ROS IN A SPECIFIED DIRECTION
```

```
C
```

```
INCLUDE 'WND.COM.INC'
```

```
INCLUDE 'QSPREAD.INC'
```

```
C
```

```
C .... INPUT ....
```

```
C SDIR — DIRECTION FOR THE CALCULATIONS
```

```
C XRATE,EWIND,XDIR FOR THE DIRECTION OF MAXIMUM SPREAD FROM VECTOR
```

```
C
```

```
C .... OUTPUT ....
```

```
C XRATE,EWIND,XBYRAM,XFLAME,LWIND IN THE DIRECTION SDIR.
```

```
C
```

```
DATA PI/3.14159/
```

```
C
```

```
IF (EWIND.EQ.0.) GO TO 999
```

```
RLW=1+.25*(EWIND/88.)
```

```
EL=SQRT(RLW**2-1.)/RLW
```

```
DIR=ABS(XDIR-SDIR)
```

```
IF (DIR.GT.180.) DIR=360.-DIR
```

```
TH=DIR*PI/180.
```

```
XRATE=XRATE*(1.-EL)/(1.-EL*COS(TH))
```

```
PHIEW=XRATE/RATE0-1.
```

```
IF (PHIEW.LT.0.) PHIEW=0.
```

```
EWIND = (PHIEW*(BETA/BETAOP)**E /C)**(1./B)
```

```
IF (EWIND.LE.WLIM) THEN
```

```
    LWIND=0
```

```
ELSE
```

```
    LWIND=1
```

```
    EWIND=WLIM
```

```
    PHIEW=C*WLIM**8*(BETA/BETAOP)**(-E)
```

```
    XRATE=XIR*XI*(1.+PHIEW)/RBQIG
```

```
ENDIF
```

```
XBYRAM=384.*XIR*XRATE/(60.*SIGMA)
```

```
XFLAME=.45*XBYRAM**46
```

```
C
```

```
IF (XBYRAM .GT. 0.) THEN
```

```
    XSCOR77= (XBYRAM**1.16666)/(XBYRAM + (EWIND/88.))**3.))**0.5
```

```
ELSE
```

```
    XSCOR77 = 0.
```

```
ENDIF
```

```
999 CONTINUE
```

```
RETURN
```

```
END
```

```
SUBROUTINE HLPWIN(HLPARG, FROM)
```

```
C... HELP SUBROUTINE FOR WIND FUNCTIONS
```

```
C...
```

```
INCLUDE 'WINDCOM.INC'
```

```
C...
```

```
INCLUDE 'IOCOM.INC'
```

```
C...
```

```
PARAMETER (NKEYS=18)
```

```
CHARACTER*(*) FROM
```

```
CHARACTER*4 ARGV2(6), ARGALL(NKEYS), HLPARG
```

```
DIMENSION N2(6), NALL(NKEYS)
```

```
DATA N2 /1,2,2,2,1,1/
```

```
DATA ARGV2 /'INPU', 'CHAN', 'LIST', 'RUN ', 'HELP', ' '/
```

```
DATA NALL /1,1,1,1,2, 3,2,3,1,1, 1,2,1,3,1, 2,3,2/
```

```
DATA ARGALL /'QUIT', 'KEY ', 'WORD', 'TERS', 'PAUS',
```

```
& 'NOPA', 'LOG ', 'NOLO', 'STAT', 'ENGL',
```

```
& 'METR', 'PERC', 'DEGR', 'COMM', 'ZOOM',
```

```
& 'REPL', 'CONT', 'LAST'/
```

```
C...
```

```
C...
```

```
C...WRITE OPENING GREETING DEFINING CALLING MODULE
```

```
WRITE(LU6,100) FROM
```

```
DO 300 I = 1,6
```

```
IF (HLPARG(1:N2(I)) .EQ. ARGV2(I)(1:N2(I)))
```

```
+ GOTO (1,2,3,4,40,50), I
```

```
300 CONTINUE
```

```
DO 350 I = 1,NKEYS
```

```
IF (HLPARG(1:NALL(I)) .EQ. ARGALL(I)(1:NALL(I)))
```

```
+ GOTO (11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28), I
```

```
350 CONTINUE
```

```
C...
```

```
C...HELP FOR A CALL WITH INVALID ARGUMENT
```

```
C...
```

```
C... HELP INVALID ARGUMENT (INSTRUCTIONS)
```

```
C...
```

```
WRITE(LU6,340)
```

```
WRITE(LU6,341)
```

```
GOTO 999
```

```
C...
```

```
C...WRITE STATEMENTS FOR TOPIC SPECIFIC HELP
```

```
C...
```

```
C... HELP FOR INPUT(1), CHANGE(2), LIST(3), RUN(4)
```

```
C...
```

```
1 WRITE(LU6,110)
```

```
GOTO 999
```

```
2 WRITE(LU6,120)
```

```
GOTO 999
```

```
3 WRITE(LU6,130)
```

```
GOTO 999
```

```
4 WRITE(LU6,140)
```

```
GOTO 999
```

```
C...
```

```
C... HELP FOR QUIT(11), KEY(12), WORDY(13), TERSE(14), PAUSE(15)
```

```
C... NOPAUSE(16), LOG(17), NOLOG(18),STATUS(19)
```

```
C...
```

```
11  ASSIGN 150 TO IFMT
    IF (FROM .EQ. 'DISPLAY') ASSIGN 155 TO IFMT
    WRITE(LU6,IFMT)
    GOTO 999
12  WRITE(LU6,170)
    GOTO 999
13  WRITE(LU6,180)
    GOTO 999
14  WRITE(LU6,190)
    GOTO 999
15  WRITE(LU6,200)
    GOTO 999
16  WRITE(LU6,210)
    GOTO 999
17  WRITE(LU6,220)
    GOTO 999
18  WRITE(LU6,230)
    GOTO 999
19  WRITE(LU6,232)
    GOTO 999
20  WRITE(LU6,233)
    GOTO 999
21  WRITE(LU6,234)
    GOTO 999
22  WRITE(LU6,235)
    GOTO 999
23  WRITE(LU6,236)
    GOTO 999
24  WRITE(LU6,237)
    GOTO 999
25  WRITE(LU6,238)
    GOTO 999
26  WRITE(LU6,239)
    GOTO 999
27  WRITE(LU6,240)
    GOTO 999
28  WRITE(LU6,241)
    GOTO 999
```

C... HELP SECTION FOR HELP ON HELP

```
40  WRITE(LU6,105)
    GOTO 999
```

C... HELP WITH NO ARGUMENTS... GIVE GENERAL GUIDES DEPENDING ON CALL
C... FROM MAIN OR DISPLAY.

```
50  CONTINUE
    IF (FROM .EQ. MAIN) THEN
```

ELSE

ENDIF

999 RETURN

C...

```
100  FORMAT(/' RXWINDOW WINDOW HELP SYSTEM -- CURRENT MODULE IS: ',A)

105  FORMAT(/
1    /' HELP      -- PROVIDES ASSISTANCE FOR MOVING THROUGH THE PROGR',
2    'AM.  FOR HELP ',
1    /'           ON THE FUNCTION OR USE OF A SPECIFIC KEYWORD, E',
2    'NTER:      ',
1    /'           HELP KEYWORD WHERE KEYWORDS IS ANY VALID RXWINDO',
2    'W KEY WORD.  ')

110  FORMAT(/
1    /' INPUT      -- STARTS A COMPLETE RXWINDOW INPUT SEQUENCE.  INPU',
2    'T IS BASED ON ',
1    /'           LINE NUMBERS WHERE:',
1    /'           LINES 1 TO 7 DESCRIBE FIRE BEHAVIOR/EFFECT CONST',
2    'RAINTS,     ',
1    /'           LINES 8 TO 15 DESCRIBE SITE CONDITIONS,          ',
1    /'           LINES 16 TO 23 DESCRIBE PRESET ENVIRONMENTAL CONS',
2    'TRAINTS, AND ',
1    /'           LINES 24 TO 26 DESCRIBE OUTPUT TABLE FORMATS.')
```

```
120  FORMAT(//' CHANGE -- ALLOWS YOU TO CHANGE A SINGLE INPUT LINE B',
1    'Y SPECIFYING THE LINE ',
2    /'           NUMBER.  ENTER LIST TO SEE LINE NUMBERS AN',
3    'D CURRENT VALUES. ',
4    /'           CHANGE IS TERMINATED BY LINE NUMBER = 0.  ')
```

```
130  FORMAT(//' LIST    -- LISTS CURRENT VALUE OF ACTIVE INPUT LINES.')
```

```
140  FORMAT(/
1    /' RUN        -- COMPUTES AND DISPLAYS PRESCRIPTION WINDOW, AND O',
2    'PTIONALLY, THE ',
1    /'           MOISTURE TABLES.  AFTER RUN, YOU WILL BE IN THE ',
2    'DISPLAY MODULE ',
1    /'           WHERE THREE NEW KEYWORDS ARE ALLOWED.  USE HELP ',
2    'FOR SPECIFIC ',
1    /'           INFORMATION ON ZOOM, REPLAY, AND CONTINUE.  AFTE',
2    'R RETURNING TO ',
1    /'           MAIN FROM THE DISPLAY MODULE, THE "LAST" SERIES ',
2    'OF KEYWORDS ARE',
1    /'           AVAILABLE TO YOU:  LAST WINDOW SHOWS THEN LAST P',
2    'RESCRIPTION   ',
1    /'           TABLE, LAST DEAD SHOWS THE LAST DEAD MOISTURE TA',
2    'BLE, AND      ',
1    /'           LAST LIVE SHOWS THE LAST LIVE MOISTURE TABLE (IF',
2    ' APPLICABLE).')
```

```
150  FORMAT(/
1    /' QUIT      -- FROM HERE, QUIT TERMINATES THE PROGRAM.')
```

```
155  FORMAT(/
1    /' QUIT      -- FROM HERE, QUIT RETURNS YOU TO THE MAIN PROGRAM.')
```

```
170  FORMAT(/
      1 /' KEY      — LISTS KEYWORDS AND THEIR EFFECT.')
```

```
180  FORMAT(/
      1 /' WORDY    — PROVIDES MAXIMUM INFORMATION WHILE PROMPTING FOR',
      2 ' RESPONSES.  ')
```

```
190  FORMAT(/
      1 /' TERSE    — PROVIDES LIMITED INFORMATION WHILE PROMPTING FOR',
      2 ' RESPONSES.  ')
```

```
200  FORMAT(/
      1 /' PAUSE    — LIMITS SCREEN DISPLAY TO 20 LINES BETWEEN CONTIN',
      2 'UE PROMPTS.')
```

```
210  FORMAT(/
      1 /' NOPAUSE  — REMOVES CONTINUE PROMPTS FROM DISPLAY.')
```

```
220  FORMAT(/
      1 /' LOG      — TURNS LOGGING FUNCTION ON.  ON FIRST LOG, YOU AR',
      2 'E ASKED FOR A ',
      1 /'          NAME FOR THE LOG FILE.  IF YOU SELECT AN EXISTIN',
      2 'G FILE, YOU ',
      1 /'          WILL HAVE THE OPTION OF RENAMING THE NEW LOG FI',
      2 'LE, OVERWRITING',
      1 /'          THE EXISTING FILE, OR APPENDING TO THE EXISTING ',
      2 'FILE.  WHEN ',
      1 /'          LOG FUNCTIONS ARE ON, LIST AND RUN OUTPUT ARE DU',
      2 'PLICATED ON THE',
      1 /'          LOG FILE FOR LATER REVIEW.')
```

```
230  FORMAT(/
      1 /' NOLOG    — TURNS LOGGING FUNCTION OFF UNTIL LOG IS ENTERED',
      2 ' AGAIN.')
```

```
232  FORMAT(/
      1 /' STATUS   — DISPLAYS STATUS OF PROGRAM MODES.')
```

```
233  FORMAT(/
      1 /' ENGLISH  — SETS ACTIVE UNIT SYSTEM TO ENGLISH.')
```

```
234  FORMAT(/
      1 /' METRIC   — SETS ACTIVE UNIT SYSTEM TO METRIC.')
```

```
235  FORMAT(/
      1 /' PERCENT  — SETS ACTIVE SLOPE UNITS TO PERCENT.')
```

```
236  FORMAT(/
      1 /' DEGREES  — SETS ACTIVE SLOPE UNITS TO DEGREES.')
```

```
237  FORMAT(/
      1 /' COMMENT  — ALLOWS YOU TO ADD NOTES OR COMMENTS TO THE LOG F',
      2 'ILE.      ')
```

```
238  FORMAT(/
      1 /' ZOOM     — ALLOWS ZOOM OR PAN OF CURRENT DISPLAY TABLE.  YO',
      2 'U ARE PROMPTED ',
      1 /'          FOR THE TABLE LIMITS AND STEP SIZE BETWEEN EACH ',
      2 'ROW AND COLUMN.')
```

```
1 /'          THE ZOOM PROMPT WILL INCLUDE VALID TABLE PARAMET',
2 'ER RANGES.          ',
1 /'          WHEN RESPONDING TO A ZOOM PROMPT, ENTER THE BEGI',
2 'NING VALUE, THE',
1 /'          END VALUE, AND THE STEP SIZE SEPARATING EACH FIE',
2 'LD WITH A COMMA.',
1 /'          E.G.  5,20,5 SHOWS VALUE AT 5,10,15, AND 20.  IN ',
2 'DISPLAYING          ',
1 /'          TABLES, THE NUMBER OF COLUMNS (STEPS) IS LIMITED',
2 ' TO 10 IN DEAD ',
1 /'          MOISTURE TABLES, AND TO 8 IN THE WINDOW AND LIVE',
2 ' MOISTURE TABLES.',
1 /'          THE ZOOM INPUT ROUTINES ALLOW YOU TO VERIFY YOUR',
2 ' SPECIFICATIONS.',
1 /'          FOR ALL TABLE TYPES, THE NUMBER OF TABLE ROWS IS',
2 ' LIMITED TO 15.')
```

239 FORMAT (/

```
1 /' REPLAY  — PERFORMS AN INSTANT REPLAY OF THE LAST DISPLAYED',
2 ' TABLE.  USE ',
1 /' ZOOM TO VIEW DIFFERENT SECTIONS OF DISPLAY TABLES.')
```

240 FORMAT(/

```
1 /' CONTINUE— CONTINUES THE LOGICAL PROGRESSION OF THE DISPLAY',
2 ' SEQUENCE:          ',
1 /'          1)HEAD FIRE RX, 2)FLANK FIRE RX, 3)BACK FIRE RX,',
2 ' 4)DEAD FUEL          ',
1 /'          MOISTURE TABLE, AND 5)LIVE FUEL MOISTURE TABLE.')
```

241 FORMAT(/

```
1 /' LAST  — ALLOWS YOU TO JUMP BACKWARDS INTO A PREVIOUSLY D',
2 ' ISPLAYED OUTPUT',
1 /'          SEQUENCE.  LAST IS ALLOWED ONLY UPON RETURNING T',
2 ' O MAIN FROM THE',
1 /'          DISPLAY MODULE.  USED IN CONJUNCTION WITH ARGUMEN',
2 ' TS "WINDOW",          ',
1 /'          "DEAD", OR "LIVE", IT REDISPLAYS THE TABLE IN ',
2 ' FULL WITHOUT          ',
1 /'          COMPUTATIONAL DELAY.  ONCE THE TABLE IS DISPLAYED',
2 ' , CONTROL          ',
1 /'          REMAINS IN THE DISPLAY MODULE WHERE YOU MAY ZOOM',
2 ' , REPLAY, OR          ',
1 /'          CONTINUE IN THE DISPLAY SEQUENCE. (EXAMPLE USEAGE',
2 ' : LAST WINDOW)')
```

340 FORMAT(/

```
&/' THIS IS THE MAIN SECTION OF "WINDOW." THE PROGRAM IS DESIGNED',
&/' TO ASSIST IN DEFINING ENVIRONMENTAL PARAMETERS THAT ALLOW A ',
&/' PRESCRIBED FIRE TO BURN WITHIN A BURNING PRESCRIPTION.'/)
```

341 FORMAT(

```
&/' PROGRAM CONTROL IS BY USE OF KEYWORDS AND THERE IS A COMPLETE',
&/' HELP SYSTEM.  IF YOU NEED MORE HELP THAN AVAILABLE FROM THE ',
&/' HELP SYSTEM, YOU MAY PRINT THE INSTRUCTION MANUAL BY ENTERING',
```

&/' THE KEYWORD "INSTRUCTIONS" AND THE XX PAGE MANUAL WILL BE ',
&/' PRINTED TO YOUR LINEPRINTER.')

280 FORMAT(/

&/' THE FIRE SECTION OF THE PROGRAM IS DESIGNED TO DISPLAY ALL ',
&/' COMBINATIONS OF EFFECTIVE WIND (WIND+SLOPE), WEIGHTED DEAD ',
&/' FUEL MOISTURE, AND WEIGHTED LIVE FUEL MOISTURE THAT RESULT IN',
&/' FIRE BEHAVIOR WITHIN THE PRESCRIPTION CONSTRAINTS. CURRENTLY',
&/' AVAILABLE CONSTRAINT PARAMETERS ARE: ',

&/' 1. RATE OF SPREAD',

&/' 2. FLAME LENGTH',

&/' 3. HEAT PER UNIT AREA',

&/' 4. FIRELINE INTENSITY',

&/' 5. REACTION INTENSITY',

&/' 6. SCORCH HEIGHT',

&/' IN ADDITION YOU MAY DEFINE ENVIRONMENTAL CONDITIONS (FUEL ',

&/' MOISTURE AND EFFECTIVE WINDSPEED) REGARDLESS OF THE FIRE BE- ',

&/' BEHAVIOR.')

C...

C...

END

```
FUNCTION IEOS(STRING)
```

```
CHARACTER*(*) STRING
```

```
C... RETURNS THE POSITION OF THE LAST NON-BLANK ELEMENT OF STRING
```

```
N = LEN(STRING)
```

```
DO 100 IEOS = N,1,-1
```

```
100 IF (STRING(IEOS:IEOS) .NE. ' ') RETURN
```

```
IEOS = 1
```

```
RETURN
```

```
END
```

SUBROUTINE INFUEL

```

C
C////PROGRAMMED BY LARRY BRADSHAW, SEM, 1986
C
C
C.... INPUT AND CHANGE FOR KEYWORD = INPUT
C.... STEPS ARE:
C.... 1. GET FUEL MODEL
C
C      INCLUDE 'WINDCOM.INC'
C
C      INCLUDE 'IOCOM.INC'
C
C... COMMENTED STATEMENTS ALLOW TWO FUEL MODELS
C
C      1 X = 999.
C      WRITE (LU6,5070)
C      5070 FORMAT (/ ' 8--FUEL MODEL ? (1-99 OR ?) ' )
C      IF (WORDY) WRITE (LU6,5080)
C... 5080 FORMAT (50H (ENTER 0 FOR TWO FUEL MODEL CONCEPT INPUT.) /
C      5080 FORMAT ( ' (ENTER 1 TO 13 FOR STANDARD NFFL MODELS, ' /
C      & ' 14 TO 99 FOR CUSTOM MODELS, OR ' /
C      & ' ? FOR A LIST OF STANDARD MODELS.) ' )
C... CALL WHAT(0.,99.,.FALSE.,.TRUE.,0,X,DUM,OK,QUIT,.TRUE.)
C      CALL WHAT(1.,99.,.FALSE.,.TRUE.,0,X,DUM,OK,QUIT,NOPE)
C      IF (.NOT.OK) GO TO 1
C      IF (QUIT) THEN
C          WRITE (LU6,5060)
C      5060 FORMAT (/ ' FUEL INPUT TERMINATED. ' )
C          GO TO 999
C      ENDIF
C...
C...CHECK FOR ? AND LIST FUEL MODELS IF NEEDED
C      8000 FORMAT(/ ' NUMBER DESCRIPTION ' )
C      8005 FORMAT(1X,I2,5X,8A4)
C      IF (X.EQ. 999) THEN
C          WRITE(LU6,8000)
C          WRITE(LU6,8005) (J,(NMOD(J,I),I=1,8),J=1,13)
C          GOTO 1
C      ENDIF
C
C      IF (X.GT.0.) THEN
C          MODL=X
C          TWO=.FALSE.
C          NTWO=1
C          CALL CHKMOD
C          IF (.NOT. OK) THEN
C              WRITE(LU6,6005)
C      6005 FORMAT(/ ' DO YOU WANT TO TRY ANOTHER MODEL NUMBER ? Y-N ' )
C          CALL READYN(YES)

```

```
      IF (YES) GOTO 1
      WRITE (LU6,9000)
9000  FORMAT(/' DO YOU WANT TO LOAD A CUSTOM FUEL MODEL ? (Y-N)')
      CALL READYN(YES)
      IF (YES) THEN
        CALL CUSTOM
        GO TO 1
      ELSE
        WRITE(LU6,9005)
9005  FORMAT(/' AT THIS POINT YOU HAVE NO CHOICE. TRY ANOTHER MODEL ',
+ 'OR LOAD A CUSTOM FILE.')
        GOTO 1
      ENDIF
    ENDIF
    VK(8,1) = MODL
    VK(8,2) = 0.
    VK(8,3) = -1.
    MP1 = 100
    IF (WORDY) WRITE(LU6,6000) MODL,(NMOD(LM1,I),I=1,8),MP1
  ELSE
    CALL READIT('(8) DOMINANT FUEL MODEL NUMBER ? 1-99 '
-      ,1.,99.,.FALSE.,.TRUE.,0,X,DUM)
    MODL=X
    TWO=.TRUE.
    NTWO=1
    CALL CHKMOD
    IF (.NOT.OK) THEN
      WRITE (LU6,9000) MODL
      CALL READYN(YES)
      IF (YES) CALL CUSTOM
      GO TO 1
    ENDIF
    M1=MODL
    CALL READIT('(8) PERCENT COVER ? 51-80 '
-      ,51.,80.,.FALSE.,.FALSE.,0,X,DUM)
    MP1=X
    CALL READIT('(8) OTHER FUEL MODEL ? 1-99 '
-      ,1.,99.,.FALSE.,.TRUE.,0,X,DUM)
    MODL=X
    NTWO=2
    CALL CHKMOD
    IF (.NOT.OK) THEN
      WRITE (LU6,9000) MODL
      CALL READYN(YES)
      IF (YES) CALL CUSTOM
      GO TO 1
    ENDIF
    VK(8,1)=M1
    VK(8,2)=MP1
    VK(8,3)=MODL
    IF (WORDY) THEN
      WRITE(LU6,6000) M1,(NMOD(LM1,I),I=1,8),MP1
      WRITE(LU6,6000) MODL,(NMOD(LM2,I),I=1,8), 100-MP1
    ENDIF
  ENDIF
```

C

```
999 CONTINUE  
RETURN  
6000 FORMAT(''  
END
```

```
FUEL MODEL: ',I2,' — ',8A4,' (',I3,'%')
```

SUBROUTINE KEY0

C

C////PROGRAMMED BY PAT ANDREWS, NFFL, 1983

C

C

C KEYWORDS FOR OPERATIONAL SECTIONS OF FIRE1

C

INCLUDE 'IOCOM.INC'

C

WRITE (LU6,1000)

1000 FORMAT (' ACTION KEYWORDS: '

```
-      /' INPUT  - ENTER ALL INPUT VALUES',
-      /' LIST   - LIST CURRENT INPUT VALUES',
-      /' CHANGE - CHANGE INPUT BY LINE NUMBER',
-      /' RUN    - DO CALCULATIONS AND PRINT RESULTS',
-      /' ZOOM   - EXPAND OR CONDENSE WINDOW OF VIEWED TABLE',
-      /' REPLAY - INSTANT REPLAY OF CURRENT TABLE',
-      /' CONTINUE- CONTINUE DISPLAY SEQUENCE',
-      /' LAST   - RE-DISPLAY AN EXISTING TABLE',
-      /' HELP   - TELLS YOU WHERE YOU ARE AND WHAT YOU CAN',
-                ' DO NEXT'
-      /' KEY    - PRINT THIS KEYWORD LIST'
-      /' COMMENT - ALLOWS YOU TO COMMENT A RUN'
-      /' QUIT   - QUIT TO NEXT PROGRAM LEVEL')
```

C

RETURN

END

SUBROUTINE KEYALL

C

C////PROGRAMMED BY PAT ANDREWS, NFFL, 1983

C

C

C PRINT KEYWORDS THAT ARE VALID FOR ANY SECTION OF FIRE1

C

INCLUDE 'IOCOM.INC'

C

CALL WAITE

WRITE (LU6,1000)

1000 FORMAT (' MODE KEYWORDS:',

- /' TERSE - FOR LIMITED PROMPTING AFTER YOU ARE USED',

-' TO THE PROGRAM'

- /' WORDY - FOR FULL PROMPTING (DEFAULT)'

- /' PAUSE - NO MORE THAN 24 LINES ARE SHOWN AT A TIM',

-'E FOR SCREEN DISPLAY'

- /' NOPAUSE - NEGATES PAUSE OPTION')

WRITE (LU6,1010)

1010 FORMAT (' LOG - COPIES LISTS AND RESULTS TO A LOG FILE'

- /' NOLOG - NEGATES LOG OPTION',

- /' STATUS - DISPLAYS CURRENT RUN PARAMTERS'

- /' ENGLISH - SETS DISPLAY TO ENGLISH UNITS'

- /' METRIC - SETS DISPLAY TO METRIC UNITS'

- /' PERCENT - SETS SLOPE UNITS TO PERCENT'

- /' DEGREES - SETS SLOPE UNITS TO DEGREES')

C

RETURN

END

SUBROUTINE KEYFLG

```
C
C////PROGRAMMED BY PAT ANDREWS, NFFL, 1983
C    MODIFIED BY ROGER L. BRADSHAW    OCTOBER 18, 1985
C    MODIFIED BY LARRY S. BRADSHAW    NOVEMBER 1986
C .... CHECK FOR KEYWORDS THAT SET OR CLEAR FLAGS WORDY AND PAUSE
C
C    INCLUDE 'WND.COM.INC'
C
C    INCLUDE 'IO.COM.INC'
C
C    OK=.FALSE.
C
C    IF (KEY(1:1).EQ.'T') THEN
C      WRITE (LU6,6000)
6000  FORMAT (25H TERSE PROMPT OPTION SET. )
C      WORDY=.FALSE.
C      OK=.TRUE.
C      GO TO 900
C    ENDIF
C
C    IF (KEY(1:1).EQ.'W') THEN
C      WRITE (LU6,6010)
6010  FORMAT (25H WORDY PROMPT OPTION SET. )
C      WORDY=.TRUE.
C      OK=.TRUE.
C      GO TO 900
C    ENDIF
C
C    IF (KEY(1:2).EQ.'PA') THEN
C      WRITE (LU6,6020)
6020  FORMAT (/ ' PAUSE OPTION SET.  YOU WILL PROMPTED TO CONTINUE.' )
C      THOUT A QUESTION, PRESS THE CARRIAGE RETURN KEY.)
C      PAUSE=.TRUE.
C      OK=.TRUE.
C      GO TO 900
C    ENDIF
C
C    IF (KEY(1:3).EQ.'NOP') THEN
C      WRITE (LU6,6030)
6030  FORMAT (21H NO-PAUSE OPTION SET. )
C      PAUSE=.FALSE.
C      OK=.TRUE.
C      GO TO 900
C    ENDIF
C
C    IF (KEY(1:2).EQ.'LO' .OR. KEY(1:3).EQ.'NOL') THEN
C      OK=.TRUE.
C      CALL LOGIT(LOGOK,LOG)
C      GOTO 900
C    ENDIF
C
C    900 CONTINUE
```

RETURN
END

```

SUBROUTINE LIFUEL(LUOUT)
C.... SUBROUTINE TO LIST FUEL MODEL(S)
C....
C      INCLUDE 'WINDCOM.INC'
C
C      INCLUDE 'IOCOM.INC'
C
C      CHARACTER*30 EXPOSE(5)
C      DATA EXPOSE /'EXPOSED','PARTIALLY SHELTERED',
& 'FULLY SHELTERED, OPEN STAND','FULLY SHELTERED, DENSE STAND',
& 'SET BY USER'/
C
10 IF (.NOT.TWO) THEN
    IF (VK(8,1).EQ.0.) THEN
        WRITE (LUOUT,6003)
6003   FORMAT (' 8--FUEL MODEL',5X,'          ** UNDECLARED ** ')
    ELSE
        M1=VK(8,1)
        WRITE (LUOUT,6005) M1,(NMOD(LM1,J),J=1,8)
6005   FORMAT (' 8--FUEL MODEL:
-           ,5X,I2,4H -- ,8A4)
    ENDIF
    ELSE
        M1=VK(8,1)
        M2=VK(8,3)
        MP1=VK(8,2)
        MP2=100-MP1
        WRITE (LUOUT,6010) MP1,M1,(NMOD(LM1,J),J=1,8),
-           MP2,M2,(NMOD(LM2,J),J=1,8)
6010   FORMAT (29H 8--TWO FUEL MODEL CONCEPT
-           ,I3,1H%,I3,4H -- ,8A4
-           /29X,I3,1H%,I3,4H -- ,8A4 )
    ENDIF
C
C... PRINT FUEL EXPOSURE INFORMATION
    NB = INDEX(EXPOSE(IEXPOS),' ')
    WRITE (LUOUT,6015) EXPOSE(IEXPOS)(:NB-1),WAF
6015  FORMAT(' 9--FUEL EXPOSURE TO WIND: ',9X,A,
+ /,37X,' (WIND ADJUSTMENT FACTOR = ',F4.2,')')
C
END
```

SUBROUTINE LOGIT(ITSOPN,ITSON)

THE PURPOSE OF THIS SUBROUTINE IS TO IMPLEMENT THE LOGGING
FILE FUNCTION. THE ROUTINE WILL CHECK TO SEE IF A LOGGING
FILE HAS ALREADY BEEN OPENED AND THE FUNCTION IS ON. IF THE
FILE IS NOT OPEN, ASK FOR A FILE NAME AND TRY TO OPEN IT.
FOUR ATTEMPTS TO OPEN A LOGGING FILE ARE MADE THEN THE USER
IS RETURNED TO THE MAIN ROUTINE.

ROGER L. BRADSHAW OCTOBER 16, 1985
MODIFIED TO ALLOW FILE REALLOCATION BY L.S. BRADSHAW, NOV 1986
ITSOPN LOGICAL VARIABLE TO DETERMINE IF A LOGGING FILE IS OPEN
ITSON LOGICAL VARIABLE TO DETERMINE IF LOGGING IS TURNED ON
FILNAM NAME USER SUPPLIES AS A LOGGING FILE

INCLUDE 'IOCOM.INC'
INCLUDE 'WINDCOM.INC'

LOGICAL ITSOPN,ITSON,THERE

CHARACTER ENDLOG*4, IANS*1
CHARACTER ALPHA(26)*1, ALPHAL(26)*1

DATA (ALPHA(I),I=1,26)/'A','B','C','D','E','F','G','H','I',
&'J','K','L','M','N','O','P','Q','R','S','T','U','V','W','X',
&'Y','Z'//,ENDLOG/'NOL'/
DATA (ALPHAL(I),I=1,26)/'a','b','c','d','e','f','g','h','i',
&'j','k','l','m','n','o','p','q','r','s','t','u','v','w','x',
&'y','z'//

IF (KEY(1:3) .NE. ENDLOG) THEN
IF (.NOT. ITSOPN) THEN

MAKE FOUR ATTEMPTS TO OPEN A LOGGING FILE

DO 50 ITRY=1,4
10 IF (ITRY .EQ. 1) THEN
WRITE(LU6,201)
ELSE
WRITE(LU6,220)
ENDIF
READ(LU5,202)LOGFIL

CHECK FOR ALPHA CHARACTER TO START FILE NAME

DO 20 I=1,26
IF (LOGFIL(1:1) .EQ. ALPHA(I) .OR.
LOGFIL(1:1) .EQ. ALPHAL(I)) GOTO 30
+
20 CONTINUE

FIRST CHARACTER NOT ALPHA. TELL THE USER AND GO TRY AGAIN.

WRITE(LU6,203)

GOTO 10

ATTEMPT TO OPEN THE SPECIFIED FILE. IF SUCCESSFUL, SAY SO AND LEAVE.

```
30      INQUIRE(FILE=LOGFIL,EXIST=THERE)
      IF (THERE) THEN
        WRITE(LU6,207) LOGFIL(1:IEOS(LOGFIL))
        READ (LU5,'(A1)') IANS
        IF (IANS .EQ. 'C') THEN
          WRITE(LU6,250)
          GOTO 50
        ENDIF
      ENDIF

      OPEN(LU4,FILE=LOGFIL,STATUS='UNKNOWN',IOSTAT=IERR,
&      ACCESS='SEQUENTIAL',ERR=40)

      IF (THERE .AND. IANS .EQ. 'A') THEN
35        READ(LU4,"(1X)",END=38)
        GOTO 35
      ENDIF

38      WRITE(LU6,204)
      ITSOPN=.TRUE.
      ITSON=.TRUE.
```

```
C
C .... WELCOME BANNER
      WRITE (LU4,6000) VRSION,DATE
6000 FORMAT(// ' PRESCRIPTION WINDOW PROGRAM: VERSION ',F3.1,' — ',A14,
-          //' DEVELOPED BY THE FIRE BEHAVIOR RESEARCH WORK UNIT, '
-          //' INTERMOUNTAIN FIRE SCIENCES LABORATORY, '
-          //' IN COOPERATION WITH SYSTEMS FOR ENVIRONMENTAL MGMT. '
-          //' MISSOULA, MONTANA')
```

GOTO 60

OPEN FAILED. SAY SO AND GO TRY AGAIN UNTIL 4 TRIES USED.

```
40      WRITE(LU6,"(' OPEN FAILURE. ERROR NO. ',I6)") IERR
50      CONTINUE
```

WE TRIED 4 TIMES SO LEAVE.

```
      WRITE(LU6,205)
      ITSOPN=.FALSE.
      ITSON=.FALSE.
      ELSE
```

COME HERE WHEN A LOGGING FILE IS OPEN AND WE GET A 'LOG' COMMAND

```
      WRITE(LU6,204)
      ITSON=.TRUE.
      ENDIF
      ELSE
```

```
C
C      COME HERE WHEN WE GET A 'NOLOG' COMMAND
C
      WRITE(LU6,200)
      ITSON=.FALSE.
      ENDIF
60 RETURN
C
C      FORMATS USED IN THIS ROUTINE
C
200 FORMAT(/' LOG IS OFF.')
```

201 FORMAT(/' WHAT FILE NAME DO YOU WANT TO USE? (12 CHARACTERS MAX)'
&/' FIRST CHARACTER MUST ALPHABETIC AND NO SPECIAL CHARACTERS'
&/' ARE ALLOWED.')

```
202 FORMAT(A12)
203 FORMAT(' THE FIRST CHARACTER MUST BE ALPHABETIC.')
```

204 FORMAT(/' LOG IS ON.')

```
205 FORMAT(/' UNABLE TO OPEN A LOGGING FILE AFTER FOUR ATTEMPTS.'  
&/' RETURNING TO CONTROL SECTION. LOG IS OFF.')
```

207 FORMAT(/' LOG "' ,A,'" EXISTS. (Append/Change/Delete)')

```
220 FORMAT(/' LOG FILE NAME?')
```

250 FORMAT(/' CHOOSE ANOTHER NAME FOR THE LOG FILE.')

```
C
      END
```

SUBROUTINE MFLAG

```
C
C////PROGRAMMED BY PAT ANDREWS, NFFL, 1983
C
C
C .... SET FLAGS TO INDICATE WHICH FUEL CLASSES HAVE LOADS
C      AND THEREFORE REQUIRE MOISTURE INPUT
C
C
C      INCLUDE 'WND.COM.INC'
C
C      INCLUDE 'IO.COM.INC'
C      INCLUDE 'UNITS.INC'
C
C      IF (NTWO.EQ.1) THEN
C          DO 100 I=1,5
C              MOI(I) = .FALSE.
100  CONTINUE
C      ENDIF
C
C      IF (TWO .AND. NTWO.EQ.2) THEN
C          LM=LM2
C      ELSE
C          LM=LM1
C      ENDIF
C
C      DO 200 I=1,5
C          IF (PMOD(LM,I).GT.0.) MOI(I) = .TRUE.
200  CONTINUE
C... SET UP NEEDED ARRAY BASED ON FUEL MODEL
C...
C      NEEDED(I1HR) = .FALSE.
C      NEEDED(I10H) = .FALSE.
C      NEEDED(IREL) = .FALSE.
C      NEEDED(IHR8) = .FALSE.
C      NEEDED(IWDY) = .FALSE.
C
C      IF (MOI(1)) NEEDED(I1HR) = .TRUE.
C      IF (MOI(2)) NEEDED(I10H) = .TRUE.
C      IF (MOI(3)) NEEDED(IREL) = .TRUE.
C      IF (MOI(4)) NEEDED(IHR8) = .TRUE.
C      IF (MOI(5)) NEEDED(IWDY) = .TRUE.
C      RETURN
C      END
```

SUBROUTINE QSPREAD(ENTER, LEAVE)

```

C-----C
C SUBROUTINE QSPREAD(ENTER, LEAVE) C
C-----C
C ARGUMENTS C
C INTEGER ENTER - Subroutine entry point (1-5); see below. C
C INTEGER LEAVE - Subroutine exit point (1-5); see below. C
C-----C
C DESCRIPTION C
C Optimized spread subroutine as used by BEHAVE. Permits partial C
C calculation of intermediates (rather than complete recalculation). C
C CALL QSPREAD(1,1) - Calculates all fuel model intermediates. C
C CALL QSPREAD(2,2) - Calculates slope intermediate constants. C
C CALL QSPREAD(3,3) - Calculates moisture intermediates. C
C CALL QSPREAD(4,4) - Calculates wind limits and intermediates. C
C CALL QSPREAD(5,5) - Calculates fire behavior results. C
C CALL QSPREAD(1,5) - Calculates everything just like SPREAD(). C
C CALL QSPREAD(1,2) - Calculates fuel model and slope constants. C
C CALL QSPREAD(3,5) - Calculates moisture, wind, and fire given C
C existing fuel model and slope constants. C
C-----C
C COMMON BLOCKS C
C /SPREADCB/ contained in QSPREAD.INC contains all common variables. C
C-----C
C LOCAL VARIABLES C
C REAL RTEMP C
C INTEGER NBOUNDS - Number of G() weight class categories. C
C REAL G(NBOUNDS) - Loading weight classes. C
C-----C
C PROGRAMMER C.D.BEVINS C
C LIBRARY C
C MODIFIED 11/25/86 C
C-----C

```

```

INCLUDE 'QSPREAD.INC'

```

```

INTEGER ENTER
INTEGER LEAVE
INTEGER I, J, N
INTEGER NBOUNDS, IBOUND
INTEGER REALS
PARAMETER (NBOUNDS = 5)
PARAMETER (REALS = 83)

```

```

SAVE

```

```

INTEGER GCLASS(NBOUNDS)
REAL BOUNDS(NBOUNDS)
REAL G(NBOUNDS)
REAL RTEMP
REAL X(REALS)
EQUIVALENCE (FUELAREA(1,1), X(1))

```

DATA BOUNDS/ 1200., 192., 96., 48., 16./

C-----
C BRANCH TO ENTRY POINT
C ENTRY PT. 1 - FUEL BED CONSTANTS
C ENTRY PT. 2 - SLOPE CONSTANTS
C ENTRY PT. 3 - MOISTURE CONSTANTS
C ENTRY PT. 4 - WIND CONSTANTS, EFFECTIVE WIND LIMIT TESTS
C ENTRY PT. 5 - FIRE BEHAVIOR CALCULATIONS
C-----

GOTO (1000,2000,3000,4000,5000), ENTER

C-----
C ENTRY PT 1: DERIVE CONSTANT FUEL MODEL INTERMEDIATES
C-----

C-----
C INITIALIZE VARIABLES TO ZERO USING EQUIVALENCED ARRAY X()
C-----

1000 DO 1010 I=1,REALS
1010 X(I) = 0.

C-----
C FIX SPECIFIC TO BEHAVE: USE LIVE FUELS ONLY IF LOAD > .001
C-----

RTEMP = FUELLOAD(HR1,LIVE) + FUELLOAD(HR10,LIVE)
N = 2
IF (RTEMP .LT. 0.001 .OR. NFC(LIVE) .LT. 1) N = 1

C-----
C DETERMINE FUEL WEIGHTING FACTORS FOR EACH FUEL CATEGORY
C-----

DO 1090 J = 1, N

C-----
C FUEL SURFACE AREA FOR EACH CLASS AND TOTAL FOR CATEGORY: FUELAREA(I,J)
C-----

DO 1020 I = 1, NFC(J)
IF (FUELDENS(I,J) .GT. 0.)
+ FUELAREA(I,J) = FUELLOAD(I,J)*FUELSAVR(I,J)/FUELDENS(I,J)
FCATAREA(J) = FCATAREA(J) + FUELAREA(I,J)
1020 CONTINUE

C-----
C FUEL CLASS WEIGHTING FACTOR WITHIN CATEGORY: FUELF(NCLASS,NCATEG)
C-----

IF (FCATAREA(J) .GT. 0.) THEN
DO 1030 I = 1, NFC(J)
1030 FUELF(I,J) = FUELAREA(I,J) / FCATAREA(J)

ENDIF

C FUEL LOADING WEIGHTING FACTOR WITHIN CATEGORY: FUELG(NCLASS,NCATEG)
C PROCEDURE ENSURES SUMMATION OF TOTAL LOAD WITHIN THE SAME SIZE CLASS

DO 1040 IBOUND = 1, NBOUNDS

1040 G(IBOUND) = 0.

DO 1070 I = 1, NFC(J)

RTEMP = FUELSAVR(I,J)

DO 1050 IBOUND = 1, NBOUNDS

1050 IF (RTEMP .GE. BOUNDS(IBOUND)) GOTO 1060

1060 G(IBOUND) = G(IBOUND) + FUELF(I,J)

GCLASS(I) = IBOUND

1070 CONTINUE

DO 1080 I = 1, NFC(J)

1080 FUELG(I,J) = G(GCLASS(I))

C NEXT FUEL CATEGORY

1090 CONTINUE

C FUEL CATEGORY WEIGHTING FACTORS: FCATF(NCATEG)

$$FCATF(DEAD) = FCATAREA(DEAD) / (FCATAREA(DEAD) + FCATAREA(LIVE))$$
$$FCATF(LIVE) = 1. - FCATF(DEAD)$$

C FUEL MOISTURE WEIGHTING FACTORS: FUELMWF(NCLASS,NCATEG)
C USED TO DERIVE MEXT OF LIVE FUELS ACCORDING TO ALBINI

DO 1100 J = 1, N

DO 1100 I = 1, NFC(J)

1100 FUELMWF(I,J) = EXP(-138./FUELSAVR(I,J))

DO 1110 I = 1, NFC(DEAD)

1110 FINED = FINED + FUELLOAD(I,DEAD) * FUELMWF(I,DEAD)

DO 1120 I = 1, NFC(LIVE)

1120 WLIVE = WLIVE + FUELLOAD(I,LIVE) * EXP(-500./FUELSAVR(I,LIVE))

IF (WLIVE .NE. 0.) WLIVE = 2.9 * FINED / WLIVE

C NOW CALCULATE FUEL BED INTERMEDIATES

DO 1140 J = 1, N

```

DO 1130 I = 1, NFC(J)
  RTEMP = FUELF(I,J)
  FCATLOAD(J) = FCATLOAD(J) + FUELG(I,J) * FUELLOAD(I,J) *
+   (1. - FUELSTOT(I,J))
  FCATHEAT(J) = FCATHEAT(J) + RTEMP * FUELHEAT(I,J)
  FCATSEFF(J) = FCATSEFF(J) + RTEMP * FUELSEFF(I,J)
  FCATSAVR(J) = FCATSAVR(J) + RTEMP * FUELSAVR(I,J)
  RHOB = RHOB + FUELLOAD(I,J)
  IF (FUELDENS(I,J) .GT. 0.)
+   BETA = BETA + FUELLOAD(I,J) / FUELDENS(I,J)
1130 CONTINUE

```

```

SIGMA = SIGMA + FCATF(J) * FCATSAVR(J)
FCATETAS(J) = AMIN1( 1.0, 0.174/(FCATSEFF(J)**0.19) )
PRI(J) = FCATLOAD(J) * FCATHEAT(J) * FCATETAS(J)
1140 CONTINUE

```

```

RHOB = RHOB / FUELDEPTH
BETA = BETA / FUELDEPTH
BETAOP = 3.348 / (SIGMA**0.8189)
RATIO = BETA / BETAOP
AA = 133. / (SIGMA**0.7913)
RTEMP = SIGMA**1.5
GAMMAX = RTEMP / (495. + 0.0594*RTEMP)
GAMMA = GAMMAX * (RATIO**AA) * EXP(AA*(1.-RATIO))
XI = EXP( (0.792 + 0.681*SIGMA**0.5) * (BETA+0.1) )
+   / (192. + 0.2595*SIGMA)

```

```

DO 1150 J = 1, N
1150 PRI(J) = GAMMA * PRI(J)

```

C WIND AND SLOPE FUEL BED CONSTANTS

```

KSLOPE = 5.275 * BETA**(-0.3)
C = 7.47 * EXP( -0.133 * SIGMA**0.55)
B = 0.02526 * SIGMA**0.54
E = 0.715 * EXP( -0.000359 * SIGMA)
KWIND = C * RATIO**(-E)

```

C END OF ENTRY PT 1 - FUEL BED CONSTANTS

```

IF (LEAVE .EQ. 1) RETURN

```

C ENTRY PT 2: SLOPE CONSTANT DERIVATION

```

2000 PHIS = KSLOPE * SLOPE * SLOPE
IF (LEAVE .EQ. 2) RETURN

```

C ENTRY PT 3: MOISTURE CONSTANT DERIVATION

3000 FCATMEXT(DEAD) = FUELMEXT
IF (N .LT. 2) GOTO 3020

C LIVE FUEL MOISTURE OF EXTINCTION

WMFD = 0.
DO 3010 I = 1, NFC(DEAD)
3010 WMFD = WMFD +
+ FUELMWF(I,DEAD) * FUELLOAD(I,DEAD) * FUELMOIS(I,DEAD)

IF (FINED .NE. 0.) FDMOIS = WMFD / FINED

FCATMEXT(LIVE) = WLIVE * (1.0 - FDMOIS/FCATMEXT(DEAD)) - 0.226
FCATMEXT(LIVE) = AMAX1(FCATMEXT(DEAD), FCATMEXT(LIVE))

C WEIGHTED FUEL MOISTURE, QIG, AND SUMMATIONS

3020 RBQIG = 0.

DO 3040 J = 1, N
FCATMOIS(J) = 0.

DO 3030 I = 1, NFC(J)
RTEMP = FUELF(I,J)
QIG(I,J) = 250. + 1116. * FUELMOIS(I,J)
FCATMOIS(J) = FCATMOIS(J) + RTEMP * FUELMOIS(I,J)
RBQIG = RBQIG + FCATF(J) * RTEMP * QIG(I,J) * FUELMWF(I,J)
3030 CONTINUE

C MOISTURE DAMPING COEFFICIENTS

RTEMP = FCATMOIS(J) / FCATMEXT(J)
FCATETAM(J) = 1. - 2.59*RTEMP + 5.11*RTEMP*RTEMP
+ - 3.52*RTEMP*RTEMP*RTEMP
IF (FCATMOIS(J) .GE. FCATMEXT(J)) FCATETAM(J) = 0.
3040 CONTINUE

XIR = 0.
DO 3050 J = 1, N
3050 XIR = XIR + PRI(J) * FCATETAM(J)

RBQIG = RHOB * RBQIG

IF (LEAVE .EQ. 3) RETURN

C ENTRY PT 4: WIND CONSTANT DERIVATION AND TESTING

4000 $PHIW = KWIND * WIND^{**B}$

$PHIEW = PHIW + PHIS$

$LWIND = 0.$

$EWIND = (PHIEW * RATIO^{**E} / C)^{**}(1./B)$

$WLIM = 0.9 * XIR$

IF (EWIND .GT. WLIM) THEN

$LWIND = 1.$

$PHIEW = KWIND * WLIM^{**B}$

$EWIND = WLIM$

ENDIF

IF (LEAVE .EQ. 4) RETURN

C ENTRY PT 5: FIRE BEHAVIOR CALCULATIONS

5000 $RATE0 = XIR * XI / RBQIG$

$RATE = RATE0 * (1.0 + PHIEW)$

$HPUA = XIR * 384. / SIGMA$

$BYRAM = HPUA * RATE / 60.$

$FLAME = 0.45 * BYRAM^{**0.46}$

IF (BYRAM .GT. 0.) THEN

$SCOR77 = (BYRAM^{**1.16666}) / (BYRAM + (WIND/88.)^{**3.})^{**0.5}$

ELSE

$SCOR77 = 0.$

ENDIF

RETURN

END

```
SUBROUTINE READIT(IQU,R1,R2,RNGOK,IGR,NVAR,VALUE,VK)
```

```
C$TEST
C
C////PROGRAMMED BY PAT ANDREWS, NFFL, 1983
C...
C *****
C * MODIFIED FOR SWITCHABLE UNITS BY *
C * LARRY BRADSHAW, SEM, 1987 *
C *****
C...
C      INCLUDE 'UNITS.INC'
C...
C
C
C ... ASK A ONE LINE QUESTION.
C... CHECK THE INPUT.
C      IF NOT OK, REASK THE QUESTION.
C      (INPUT = 'QUIT' IS NOT ALLOWED.)
C
C      DIMENSION V(3),VK(33,3)
C      LOGICAL OK,RNGOK,IGR,QUIT,QOK
C      CHARACTER*(*) IQU
C...
C.... TEST THE INPUT PROMPT FOR A COMMA. IF THERE IS ONE, MODIFY THE
C... PROMPT FOR UNITS AND RANGES. IF NO COMMA, INPUT IS NOT UNIT
C... SENSITIVE
C...
C      DO 5 I = 1,70
5 STR(I:I) = ' '
      NPOS = INDEX(IQU,',')
      IF (NPOS.NE.0) THEN
          NV = IABS(NVAR)
          IF (NV.EQ.0) NV = 1
          IUPT = IUNT
          IF (NV.EQ.ISLO) IUPT = ISLP
          LU = IEOS(UNTLBL(IUPT,ISHT,NV))
          CALL SETRNG(RLOW,R1,RHIG,R2,NV)
          LR = IEOS(RNGSTR)
          STR(1:NPOS+1) = IQU(1:NPOS+1)
          STR(NPOS+2:) = UNTLBL(IUPT,ISHT,NV)(1:LU)//
+          ' ? '//RNGSTR(1:LR)
          JPOS = INDEX(STR,'20-FOOT')
          IF (JPOS.NE.0.AND.METRIC) STR(JPOS:JPOS+6) = '6-METER'
          LS = IEOS(STR)
      ELSE
          LS = IEOS(IQU)
          STR(1:LS) = IQU(1:LS)
          RLOW = R1
          RHIG = R2
      ENDIF
C
C .... ASK THE QUESTION
10 CALL ASK2(LS,STR,.TRUE.)
C
```

C READ THE INPUT VALUES

QOK=.FALSE.

CALL VALUES(V,OK,QOK,QUIT)

IF (.NOT.OK) GO TO 10

C

C CHECK THE VALUES

CALL CHECK(V,RLOW,RHIG,RNGOK,IGR,NVAR,VALUE,VK,OK)

IF (.NOT.OK) GO TO 10

C

RETURN

END

```
SUBROUTINE READQ(IQU,R1,R2,RNGOK,IGR,NVAR,VALUE,VK,QUIT)
```

```
C$TEST
C
C////PROGRAMMED BY PAT ANDREWS, NFFL, 1983
C
C
C... ASK A ONE LINE QUESTION.
C    CHECK THE INPUT.
C    IF THE INPUT IS 'QUIT' THEN RETURN WITH QUIT=.TRUE.
C    IF NOT OK, REASK THE QUESTION.
C...
C *****
C *   MODIFIED FOR SWITCHABLE UNITS BY           *
C *   LARRY BRADSHAW, SEM, 1987                 *
C *****
C...
C    INCLUDE 'UNITS.INC'
C...
C
C    DIMENSION V(3),VK(33,3)
C    LOGICAL OK,RNGOK,IGR,QUIT,QOK
C    CHARACTER*(*) IQU
C...
C.... TEST THE INPUT PROMPT FOR A COMMA.  IF THERE IS ONE, MODIFY THE
C... PROMPT FOR UNITS AND RANGES.  IF NO COMMA, INPUT IS NOT UNIT
C... SENSITIVE
C...
C    DO 5 I = 1,70
5 STR(I:I) = ' '
NPOS = INDEX(IQU,', ')
IF (NPOS.NE. 0) THEN
    NV = IABS(NVAR)
    IF (NV.EQ. 0) NV = NON
    IUPT = IUNT
    IF (NV.EQ. ISLO) IUPT = ISLP
    LU = IEOS(UNTLBL(IUPT,ISHT,NV))
    CALL SETRNG(RLOW,R1,RHIG,R2,NV)
    LR = IEOS(RNGSTR)
    STR(1:NPOS+1) = IQU(1:NPOS+1)
    STR(NPOS+2:) = UNTLBL(IUPT,ISHT,NV)(1:LU)//
+    ' ? '//RNGSTR(1:LR)//' OR QUIT'
    LS = IEOS(STR)
ELSE
    LS = IEOS(IQU)
    STR(1:LS) = IQU(1:LS)
    RLOW = R1
    RHIG = R2
ENDIF
C
C.... ASK THE QUESTION
10 CALL ASK2(LS,STR,.TRUE.)
C
C.... READ THE INPUT VALUES
QOK=.TRUE.
```

```
CALL VALUES(V,OK,QOK,QUIT)
```

```
IF (QUIT) GO TO 999
```

```
IF (.NOT.OK) GO TO 10
```

```
C
```

```
C . . . . CHECK THE VALUES
```

```
CALL CHECK(V,RLOW,RHIG,RNGOK,IGR,NVAR,VALUE,VK,OK)
```

```
IF (.NOT.OK) GO TO 10
```

```
C
```

```
999 CONTINUE
```

```
RETURN
```

```
END
```

SUBROUTINE READYN(YES)

C

C////PROGRAMMED BY PAT ANDREWS, NFFL, 1983

C

C

C READ THE ANSWER TO A YES-NO QUESTION.

C FOR A BAD ANSWER, GIVE THE USER ANOTHER TRY.

C

INCLUDE 'IOCOM.INC'

C

CHARACTER*1 LY, LN, L, UY, UN

LOGICAL YES

C

DATA UY/'Y'/, UN/'N'/, LY/'y'/, LN/'n'/'

C

10 READ(LU5,1000) L

1000 FORMAT (A1)

IF (L.NE.LY .AND. L.NE.LN .AND. L.NE.UY .AND. L.NE.UN) THEN

WRITE (LU6,1010) L

1010 FORMAT (1X,A1,25H IS NOT A VALID ANSWER.

- /30H TYPE Y FOR YES OR N FOR NO.)

GO TO 10

ENDIF

C

IF (L.EQ.LY .OR. L.EQ.UY) YES=.TRUE.

IF (L.EQ.LN .OR. L.EQ.UN) YES=.FALSE.

C

RETURN

END

```
SUBROUTINE SETRNG(RNGLOW,RLOW,RNGHIG,RHIG,IVAR)
```

```
C...THIS ROUTINE
```

```
C      1. CLEARS RNGSTR
C      2. CONVERTS UPPER AND LOWER RANGE LIMITS IF NEEDED
C      3. BUILDS A STRING WITH THE RANGES USING RNGSTR AS
C          AN INTERNAL FILE
```

```
INCLUDE 'UNITS.INC'
```

```
CHARACTER*4 IFORM(6), IFMT(7)
```

```
DIMENSION VAL(7)
```

```
DATA VAL /0.,1.0,10.,100.,1000.,10000.,100000./
```

```
DATA IFMT /'F2.1','I1 ','I2 ','I3 ','I4 ','I5 ','I6 '/
```

```
DATA IFORM /' (' ',' ',4H ', 'TO ', ' ', ' ') '/
```

```
C...CLEAR STRING
```

```
RNGSTR(1:20) = ' '
```

```
C...SET RANGES
```

```
RNGLOW = CVRTU(RLOW,CFAC(IVAR),METRIC,NOT2BA)
```

```
IF (RNGLOW .GT. 0. .AND. RNGLOW .LT. 0.1) RNGLOW = 0.1
```

```
IF (RNGLOW .GT. 1.0) RNGLOW = FLOAT(NINT(RNGLOW))
```

```
IF (RNGLOW .GE. 0.1 .AND. RNGLOW .LT. 1.0) THEN
```

```
  I = NINT(10.*RNGLOW)
```

```
  RNGLOW = FLOAT(I)/10.
```

```
ENDIF
```

```
RNGHIG = FLOAT(NINT(CVRTU(RHIG,CFAC(IVAR),METRIC,NOT2BA)))
```

```
C...SET FORMAT SPECS
```

```
IFORM(2) = IFMT(7)
```

```
IFORM(5) = IFMT(7)
```

```
DO 10 I = 1,6
```

```
  IF (RNGLOW .GE. VAL(I) .AND. RNGLOW .LT. VAL(I+1) )
```

```
+    IFORM(2) = IFMT(I)
```

```
  IF (RNGHIG .GE. VAL(I) .AND. RNGHIG .LT. VAL(I+1) )
```

```
+    IFORM(5) = IFMT(I)
```

```
10 CONTINUE
```

```
IF (RNGLOW .EQ. 0.0) IFORM(2) = IFMT(2)
```

```
C...ENCODE TO A CHARACTER STRING
```

```
IF (RNGLOW .GT. 0.0 .AND. RNGLOW .LT. 1.) THEN
```

```
  WRITE(RNGSTR,IFORM) RNGLOW,NINT(RNGHIG)
```

```
ELSE
```

```
  WRITE(RNGSTR,IFORM) NINT(RNGLOW),NINT(RNGHIG)
```

```
ENDIF
```

```
RETURN
```

```
END
```

LOGICAL FUNCTION SETUNT

```

C...
C... ROUTINE TO CONTROL UNIT SELECTION AND DISPLAY CURRENT UNITS
C...
      INCLUDE 'UNITS.INC'
C...
      INCLUDE 'IOCOM.INC'
      INCLUDE 'WINDCOM.INC'
      LOGICAL INTERN
C
      PARAMETER (MODULS = 1)
      CHARACTER NAMOD(MODULS)*15, PARM(MODULS,16)*25, TYPEU(2)*7,
+     TYPES(2)*7, WORD*4
      DIMENSION NPARM(MODULS), IUNPT(MODULS,16)
      DATA TYPEU /'ENGLISH', 'METRIC'/, TYPES /'PERCENT', 'DEGREES'/
      DATA NAMOD /'RXWINDOW'/
      DATA NPARM /12/
      DATA (IUNPT(1,J),J=1,16) /1,2,3,4,5,6,7,8,10,12,13,14,15,16,20,22/
C...DIRECT
      DATA (PARM(1,J),J=1,16)
+/'SPREAD RATE', 'HEAT PER UNIT AREA', 'FLAME LENGTH',
+ 'FIRELINE INTENSITY', 'REACTION INTENSITY', 'SCORCH HEIGHT',
+ 'MORTALITY', 'FUEL MODEL',
+ 'TERRAIN SLOPE', 'TREE SPECIES', 'TREE DBH', 'TREE HEIGHT',
+ 'CROWN RATIO',
+ 'FUEL MOISTURES', 'WIND SPEED', 'WIND & SPREAD DIRECTION'/

      WORD = KEY
      SETUNT = .FALSE.
      INTERN = .FALSE.

1000 FORMAT(/' CURRENT UNITS SYSTEM: ',A7,'.'3X,'SLOPE IS IN ',A7)
1005 FORMAT(T30,' UNITS SYSTEM FOR: ',A,
      A      // T33,' ENGLISH ',5X,' METRIC ',
      B      /T33,'-----',5X,'-----')
1010 FORMAT(5X,A25,A15,5X,A15)
1015 FORMAT(/' UNITS KEYWORD ?')
1020 FORMAT(' ENGLISH,METRIC,PERCENT,DEGREES,LIST,QUIT' )
5000 FORMAT(A4)

C...TEST KEYWORD
40 IF (KEY(1:1) .EQ. 'U') THEN
      INTERN = .TRUE.
      SETUNT = .TRUE.
50  WRITE(LU6,1015)
      IF (WORDY) WRITE(LU6,1020)
      READ(LU5,5000) WORD
      IF (WORD(1:1) .EQ. 'E') GOTO 100
      IF (WORD(1:1) .EQ. 'M') GOTO 200
      IF (WORD(1:1) .EQ. 'P') GOTO 300
      IF (WORD(1:1) .EQ. 'D') GOTO 400
      IF (WORD(1:1) .EQ. 'Q') GOTO 500
      IF (WORD(1:1) .EQ. 'L') THEN
          DO 70 I = 1,MODULS

```

```
        WRITE(LU6,1005) NAMOD(I)
        DO 60 J = 1,NPARM(I)
            IP = IUNPT(I,J)
            IF (IP.EQ.ISLO) THEN
                WRITE(LU6,1010) PARM(I,J),'% OR DEG ',
+                '% OR DEG '
            ELSE
                WRITE(LU6,1010) PARM(I,J),UNTLBL(IBAS,LONG,IP),
+                UNTLBL(IMET,LONG,IP)
            ENDIF
        60 CONTINUE
C      GOTO 50
    70 CONTINUE
        ENDIF
        GOTO 50
    ENDIF
C...
    100 IF (WORD(1:1) .EQ. 'E') THEN
        METRIC = .FALSE.
        IUNT = IBAS
        SETUNT = .TRUE.
    ENDIF
C...
    200 IF (WORD(1:1) .EQ. 'M') THEN
        METRIC = .TRUE.
        IUNT = IMET
        SETUNT = .TRUE.
    ENDIF
C...
    300 IF (WORD(1:2) .EQ. 'PE') THEN
        PERCENT = .TRUE.
        ISLP = IPCT
        SETUNT = .TRUE.
    ENDIF
C...
    400 IF (WORD(1:1) .EQ. 'D') THEN
        PERCENT = .FALSE.
        ISLP = IDEG
        SETUNT = .TRUE.
    ENDIF
    500 IF (SETUNT) WRITE(LU6,1000) TYPEU(IUNT),TYPES(ISLP)
        IF (INTERN .AND. WORD(1:1) .NE. 'Q') GOTO 40
        RETURN
    END
```

SUBROUTINE SETUP

```
C
C////PROGRAMMED BY PAT ANDREWS, NFFL, 1983
C
C
C .... ASSIGN VARIABLES REQUIRED BY SUBROUTINE SPREAD
C
C
C   INCLUDE 'WND.COM.INC'
C
C   INCLUDE 'SP.COM.INC'
C
C   INCLUDE 'QS.SPREAD.INC'
C
C   SAVE M
C
C   IF (MODL.LE.13) THEN
C     M=MODL
C   ELSE
C     IF (NTWO.EQ.1) THEN
C       M=14
C     ELSE
C       M=15
C     ENDIF
C   ENDIF
C
C .... FUEL MODEL PARAMETERS
C
C   FUELLOAD(1,1) = PMOD(M,1)
C   FUELLOAD(2,1) = PMOD(M,2)
C   FUELLOAD(3,1) = PMOD(M,3)
C   FUELLOAD(1,2) = PMOD(M,4)
C   FUELLOAD(2,2) = PMOD(M,5)
C   FUELSAVR(1,1) = PMOD(M,6)
C   FUELSAVR(2,1) = SIG(1,2)
C   FUELSAVR(3,1) = SIG(1,3)
C   FUELSAVR(1,2) = PMOD(M,7)
C   FUELSAVR(2,2) = PMOD(M,8)
C   FUELHEAT(1,1) = PMOD(M,10)
C   FUELHEAT(2,1) = PMOD(M,10)
C   FUELHEAT(3,1) = PMOD(M,10)
C   FUELHEAT(1,2) = PMOD(M,10)
C   FUELHEAT(2,2) = PMOD(M,10)
C   FUELDEPTH = PMOD(M,9)
C   FUELMEXT = PMOD(M,11)
C   NFC(1) = NCLAS(1)
C   NFC(2) = NCLAS(2)
C   DO 100 I = 1,3
C     DO 100 J = 1,2
C       FUELSTOT(I,J) = STOT(J,I)
C       FUELSEFF(I,J) = SEFF(J,I)
C       FUELDENS(I,J) = DENS(J,I)
C   100 CONTINUE
C   IF (WAF .EQ. 0) WAF = PMOD(M,13)
```

```
C
C... TRANSFER LOAD FOR DYNAMIC MODELS
C
C... ENTRY POINT FOR REPEAT RUNS
C...
    ENTRY SETMOI

    IF (PMOD(M,12).EQ.1) THEN
        FCC=-.0111*FM(4)+1.33
        IF (FCC.GT.1.) FCC=1.
        IF (FCC.LT.0.) FCC=0.
        OVER = FUELLOAD(2,1)*FCC
        FUELLOAD(1,1) = FUELLOAD(1,1)+OVER
        FUELLOAD(2,1) = FUELLOAD(2,1)-OVER
    ENDIF

C
C... ENVIRONMENTAL INPUT
C
C .... CHANGE WINDSPEED FROM MPH TO FT/MIN
    WIND = WMID*88.

C
C .... CHANGE SLOPE FROM PERCENT TO TANGENT
    SLOPE = PSLOP*.01

C
C .... CHANGE MOISTURE FROM PERCENT TO FRACTION
    FUELMOIS(1,1) = FM(1)*.01
    FUELMOIS(2,1) = FM(2)*.01
    FUELMOIS(3,1) = FM(3)*.01
    FUELMOIS(1,2) = FM(4)*.01
    FUELMOIS(2,2) = FM(5)*.01

C
    RETURN
END
```

```
FUNCTION SETWAF(ETW)
  INCLUDE 'IOCOM.INC'
  INCLUDE 'WINDCOM.INC'
  IEXPOS = NINT(ETW)
  IF (ETW.EQ.1.) WF=PMOD(LM1,13)
  IF (ETW.EQ.2.) WF=.3
  IF (ETW.EQ.3.) WF=.2
  IF (ETW.EQ.4.) WF=.1
  IF (ETW.EQ.5.) CALL READIT(
-    ' WIND REDUCTION FACTOR (.1 - 1.0)',
-    .1,1.0,.FALSE.,.FALSE.,0,WF,DUM)
  WRITE (LU6,5280) WF
5280 FORMAT (/ '      WIND ADJUSTMENT FACTOR = ',F4.1)
  SETWAF = WF
  RETURN
END
```

```
SUBROUTINE STATUS(FROM,PROMPT,SCREEN,LOGON,LOGOPE,FFILE)
```

```
C..
```

```
C... STATUS SUBROUTINE
```

```
C..
```

```
INCLUDE 'IOCOM.INC'
```

```
C... 
```

```
INCLUDE 'UNITS.INC'
```

```
C... 
```

```
LOGICAL PROMPT,SCREEN,LOGON,LOGOPE,FFILE
```

```
CHARACTER*4 FROM
```

```
CHARACTER*16 WHERE(5),HEAD(8),WHAT
```

```
DATA HEAD /'ACTIVE MODULE : ','PROMPT MODE : ',
```

```
& 'DISPLAY MODE : ','LOG FILE NAME : ','LOG FUNCTIONS : ',
```

```
& 'FUEL FILE NAME: ','DISPLAY UNITS : ','SLOPE UNITS : '/
```

```
DATA WHERE /'RXWINDOW (MAIN) ','DISPLAY ',
```

```
& 'WIND ','FUEL MODEL ','FIRE BEHAVIOR ' /
```

```
C... 
```

```
IF (FROM .EQ. 'MAIN') IFROM = 1
```

```
IF (FROM .EQ. 'DISP') IFROM = 2
```

```
IF (FROM .EQ. 'WIND') IFROM = 3
```

```
IF (FROM .EQ. 'FUEL') IFROM = 4
```

```
IF (FROM .EQ. 'FIRE') IFROM = 5
```

```
C... 
```

```
C...WRITE OPENING GREETING DEFINING CALLING MODULE
```

```
WRITE(LU6,100)
```

```
100 FORMAT(/' **** WINDOW STATUS REQUEST ****'/)
```

```
WRITE (LU6,105) HEAD(1),WHERE(IFROM)
```

```
105 FORMAT(5X,A16,1X,A15)
```

```
C... 
```

```
WHAT = 'TERSE'
```

```
IF (PROMPT) WHAT = 'WORDY'
```

```
WRITE(LU6,105) HEAD(2),WHAT
```

```
C... 
```

```
WHAT = 'NOPAUSE'
```

```
IF (SCREEN) WHAT = 'PAUSE'
```

```
WRITE(LU6,105) HEAD(3),WHAT
```

```
C... 
```

```
WHAT = 'UNDECLARED'
```

```
IF (LOGOPE) WHAT = LOGFIL
```

```
WRITE(LU6,105) HEAD(4),WHAT
```

```
C... 
```

```
WHAT = 'OFF'
```

```
IF (LOGON) WHAT = 'ON'
```

```
WRITE(LU6,105) HEAD(5),WHAT
```

```
C... 
```

```
WHAT = 'UNDECLARED'
```

```
IF (FFILE) WHAT = FILNAM
```

```
WRITE(LU6,105) HEAD(6),WHAT
```

```
C... 
```

```
WHAT = 'ENGLISH'
```

```
IF (METRIC) WHAT = 'METRIC'
```

```
WRITE(LU6,105) HEAD(7),WHAT
```

```
C... 
```

```
WHAT = 'PERCENT'
```

```
IF (.NOT. PERCENT) WHAT = 'DEGREES'  
WRITE(LU6,105) HEAD(8), WHAT
```

```
C...
```

```
RETURN  
END
```

```
SUBROUTINE VALUES(V,OK,QOK,QUIT)
```

```
C
```

```
C////PROGRAMMED BY PAT ANDREWS, NFFL, 1983
```

```
C... MODIFIED BY LSB TO ALLOW NEGATIVE VALUES AND '?'
```

```
C
```

```
C.... READ ALPHA INPUT CHARACTER BY CHARACTER AND CONVE RT TO
```

```
C
```

```
  A NUMERIC VALUE.
```

```
C
```

```
  DECIMAL POINT NOT REQUIRED.
```

```
C
```

```
  ACCURACY ONLY TO TENTHS.
```

```
C
```

```
  NO IMBEDDED BLANKS.
```

```
C
```

```
  SINGLE VALUE OR THREE VALUES SEPARATED BY COMMAS.
```

```
C
```

```
  NON-NEGATIVE VALUES.
```

```
C
```

```
  IF QOK=.TRUE. THEN INPUT = 'QUIT' IS OK
```

```
C
```

```
  IF INPUT IS 'QUIT' THEN QUIT=.TRUE. AND RETURN.
```

```
C
```

```
  INCLUDE 'IOCOM.INC'
```

```
C
```

```
  DIMENSION K(21),M(13),V(3),Q(4),QL(4)
```

```
  LOGICAL OK,QOK,QUIT,NEG
```

```
C
```

```
  CHARACTER*1 M,K,Q,QL, QM
```

```
C
```

```
  DATA (M(J),J=1,13)/'0','1','2','3','4','5','6','7','8','9',
```

```
-      ' ',' ',' ',' ',' '/
```

```
-      K(21)/' '/
```

```
-      (Q(I),I=1,4)/'Q','U','I','T',
```

```
-      (QL(I),I=1,4)/'q','u','i','t',
```

```
  DATA QM /'?'/
```

```
C
```

```
  QUIT = .FALSE.
```

```
  NEG = .FALSE.
```

```
  OK = .TRUE.
```

```
  V(1) = -1.
```

```
  V(2) = -1.
```

```
  V(3) = -1.
```

```
C
```

```
  DO 10 I=1,20
```

```
  K(I) = ' '
```

```
10 CONTINUE
```

```
C
```

```
  READ(LU5,1000) (K(I),I=1,20)
```

```
1000 FORMAT (20A1)
```

```
C
```

```
C... TEST FOR QUESTION MARK (RETURN WITH V(1) .EQ. 999)
```

```
  IF (K(1) .EQ. QM) THEN
```

```
    V(1) = 999.
```

```
    RETURN
```

```
  ENDIF
```

```
  IF (QOK) THEN
```

```
    DO 15 I=1,4
```

```
15     IF (K(I).NE.Q(I) .AND. K(I) .NE. QL(I) ) GO TO 18
      QUIT=.TRUE.
      GO TO 999
      ENDIF
C
C... DECIPHER INPUT STRING AND PUT VALUES IN V(1,2,3)
C...
18  I=1
    L=1
C... CHECK FOR NEGATIVE RELATION OF 100 HOUR FM TO 10 HOUR FM
    IF (K(1) .EQ. '-') THEN
      NEG = .TRUE.
      I = 2
    ENDIF
20  CONTINUE
    DO 30 J=1,10
      IF (K(I).EQ.M(J)) GO TO 35
30  CONTINUE
    GO TO 50
35  V(L) = FLOAT(J-1)
37  I=I+1
    DO 40 J=1,10
      IF (K(I).EQ.M(J)) GO TO 45
40  CONTINUE
    GO TO 50
45  V(L) = 10.*V(L) + FLOAT(J-1)
    GO TO 37
50  IF (K(I).NE.M(11)) GO TO 90
    I=I+1
    DO 60 J=1,10
      IF (K(I).EQ.M(J)) GO TO 65
60  CONTINUE
    GO TO 90
65  IF (V(L).LT.0.) THEN
      V(L) = .1*FLOAT(J-1)
    ELSE
      V(L) = V(L) + .1*FLOAT(J-1)
    ENDIF
    I=I+1
90  IF (V(L).LT.0.) GO TO 100
    IF (L.NE.2 .AND. K(I).EQ.M(12)) GO TO 999
C... ALLOW FOR TWO VALUES ONLY...INC'
    IF (L.NE.3 .AND. K(I).EQ.M(12)) THEN
      V(3) = V(2)-V(1)
      GOTO 999
    ENDIF
C...
    IF (L.NE.3 .AND. K(I).EQ.M(13)) THEN
      I=I+1
      L=L+1
      GO TO 20
    ENDIF
100 OK = .FALSE.
    WRITE (LU6,2000) (K(I),I=1,20)
2000 FORMAT (/17H ILLEGAL INPUT: ,20A1
```

```
-      /12H TRY AGAIN.  )  
999 CONTINUE  
  IF (NEG) THEN  
    V(1) = -1.*V(1)  
    V(2) = 0.  
    V(3) = -1.  
  ENDIF  
  RETURN  
END
```

```
SUBROUTINE VECTOR(XDIR,XRATE,XBYRAM,XFLAME,XSCOR77)
```

C

```
C////PROGRAMMED BY PAT ANDREWS, NFFL, 1983
```

```
C    MODIFIED BY LARRY BRADSHAW, SEM, 1988 FOR RXWINDOW APPLICATION
```

C

```
C    .... FIND THE DIRECTION OF MAXIMUM SPREAD
```

C

```
    INCLUDE 'WND.COM.INC'
```

```
    INCLUDE 'QSPREAD.INC'
```

C

```
    DATA PI/3.14159/
```

```
    IF (RATE0 .LE. 0.) THEN
```

```
        XRATE = 0.
```

```
        XBYRAM = 0.
```

```
        XFLAME = 0.
```

```
        XSCOR77 = 0.
```

```
        RETURN
```

```
    ENDIF
```

C

```
    OM=WDIR*PI/180.
```

```
    SV=RATE0*PHIS
```

```
    WV=RATE0*PHIW
```

```
    X=SV+WV*COS(OM)
```

```
    Y=WV*SIN(OM)
```

```
    RV=SQRT(X*X+Y*Y)
```

```
    IF (RV .EQ. 0) RV = 1
```

```
    AL=ASIN(ABS(Y)/RV)
```

```
    IF (X.GE.0. .AND. Y.GE.0.) A=AL
```

```
    IF (X.LT.0. .AND. Y.GE.0.) A=PI-AL
```

```
    IF (X.LT.0. .AND. Y.LT.0.) A=PI+AL
```

```
    IF (X.GE.0. .AND. Y.LT.0.) A=2.*PI-AL
```

```
    XDIR=A*180./PI
```

```
    XRATE=RV+RATE0
```

```
    PHIEW=XRATE/RATE0-1.
```

```
    EWIND = (PHIEW*(BETA/BETAOP)**E /C)**(1./B)
```

```
    IF (EWIND.LE.WLIM) THEN
```

```
        LWIND=0
```

```
    ELSE
```

```
        LWIND=1
```

```
        EWIND=WLIM
```

```
        PHIEW=C*WLIM**B*(BETA/BETAOP)**(-E)
```

```
        XRATE=XIR*XI*(1.+PHIEW)/RBQIG
```

```
    ENDIF
```

```
    XBYRAM=384.*XIR*XRATE/(60.*SIGMA)
```

```
    XFLAME=.45*XBYRAM**0.46
```

C

```
    IF (XBYRAM .GT. 0.) THEN
```

```
        XSCOR77= (XBYRAM**1.16666)/ (XBYRAM + (EWIND/88.)*3.)*0.5
```

```
    ELSE
```

```
        XSCOR77 = 0.
```

```
    ENDIF
```

```
    RETURN
```

```
    END
```


SUBROUTINE WAITE

C

C////PROGRAMMED BY PAT ANDREWS, NFFL, 1983

C

C

C PAUSE FOR SMALL CRT SCREENS---CONTINUE WITH USER RETURN

C

CALLED WHEN PAUSE = .TRUE.

C

INCLUDE 'WND.COM.INC'

C

INCLUDE 'I.COM.INC'

C

C

IF (PAUSE) THEN

WRITE(LU6, "(/' PRESS ENTER TO CONTINUE....'")

READ (LU5, 1010)

ENDIF

1010 FORMAT (A1)

RETURN

END

```
SUBROUTINE WHAT(R1,R2,RNG,IGR,NVAR,VALUE,VK,OK,QUIT,QOK)
```

```
C
```

```
C////PROGRAMMED BY PAT ANDREWS, NFFL, 1983
```

```
C
```

```
C
```

```
C .... CHECK INPUT VALUES
```

```
C     INPUT = 'QUIT' IS ALLOWED IF QOK IS .TRUE., ELSE NOT ALLOWED
```

```
C
```

```
    DIMENSION V(3),VK(33,3)
```

```
    LOGICAL OK,RNG,IGR,QUIT,QOK
```

```
C
```

```
C .... READ THE INPUT VALUES
```

```
    CALL VALUES(V,OK,QOK,QUIT)
```

```
    IF (QUIT) GO TO 999
```

```
    IF (.NOT.OK) GO TO 999
```

```
C
```

```
C .... CHECK THE VALUES
```

```
    IF (V(1) .NE. 999.) CALL CHECK(V,R1,R2,RNG,IGR,NVAR,VALUE,VK,OK)
```

```
C
```

```
999 CONTINUE
```

```
C
```

```
    RETURN
```

```
    END
```

```
SUBROUTINE ZERO
```

```
C... ZERO MOISTURE ARRAY, MSD  
INCLUDE 'DISPLAY.INC'
```

```
DO 10 I = 1,MXF
```

```
DO 10 J = 1,MXP
```

```
MSD(I,J,1) = 999
```

```
10 MSD(I,J,2) = -999
```

```
RETURN
```

```
END
```